Kausalya R

1.

|  |
| --- |
|  |
|  |  |
|  | public class q1 { |
|  | public static void main(String[] args) { |
|  | int arg1 = Integer.parseInt(args[0]); |
|  | int arg2 = Integer.parseInt(args[1]); |
|  | int arg3 = Integer.parseInt(args[2]); |
|  | if( arg1 == arg2 && arg2 == arg3) |
|  | System.out.println("equal"); |
|  | else |
|  | System.out.println("not equal"); |
|  | return; |
|  | } |
|  | } |

2.

|  |
| --- |
|  |
|  | import java.util.Arrays; |
|  | public class q2 { |
|  | public static void sort\_binary\_nums(int[] b\_nums) |
|  | { |
|  | int k = 0; |
|  |  |
|  | for (int i = 0; i < b\_nums.length; i++) |
|  | { |
|  | if (b\_nums[i] == 0) { |
|  | b\_nums[k++] = 0; |
|  | } |
|  | } |
|  |  |
|  | for (int i = k; i < b\_nums.length; i++) { |
|  | b\_nums[k++] = 1; |
|  | } |
|  | } |
|  |  |
|  | public static void main (String[] args) |
|  | { |
|  | int b\_nums[] = { 0, 1, 1, 0, 1, 1, 0, 1, 0, 0 }; |
|  | System.out.println("Original array: "+Arrays.toString(b\_nums)); |
|  | sort\_binary\_nums(b\_nums); |
|  | System.out.println("After sorting: "+Arrays.toString(b\_nums)); |
|  |  |
|  | } |
|  | } |

3.

|  |
| --- |
|  |
|  |  |
|  | public class q3 { |
|  | public static void main(String[] args) { |
|  |  |
|  | // read in one command-line argument |
|  | int n = Integer.parseInt(args[0]); |
|  |  |
|  | int i = 0; // count from 0 to N |
|  | int powerOfTwo = 1; // the ith power of two |
|  |  |
|  | // repeat until i equals n |
|  | while (i <= n) { |
|  | System.out.println(i + " " + powerOfTwo); // print out the power of two |
|  | powerOfTwo = 2 \* powerOfTwo; // double to get the next one |
|  | i = i + 1; |
|  | } |
|  |  |
|  | } |
|  | } |

4.

|  |
| --- |
|  |
|  | ++i returns the value after it is incremented, while i++ return the value before it is incremented. |
|  | At the end, in both cases the i will have its value incremented. |
|  |  |

5.

|  |
| --- |
|  |
|  | The condition is a boolean expression that tests the loop control variable. |
|  | The iteration is used to increment or decrement the loop control variable. |
|  | All three initialization, condition and iteration are optional. i.e. we can have for loop without initialization section or condition section or iteration section. |
|  | "Intialization" section is called only once before the for for starts. i.e. No matter how many iterations are there, it is executed only once at the beginning of the for loop. That is why any declarations and intializations, which we do only once goes in this section. |
|  | "Condition" is checked every time before starting the iteration, and only if it is true the body is executed, else the for loop is terminated. |
|  | "Iteration" is called at the end of the body after every iteration. So if there are 5 iterations, the iteration section is executed 5 times. |

6.

|  |
| --- |
|  |
|  | Yes we can use double variables in for loop  Especially if we want to loop through fractional values or increment by fractional values. |

7.

|  |  |
| --- | --- |
|  | All for loops can be written as while loops, and vice-versa. |
|  |  |
|  | In general, we should use a for loop when we know how many times the loop should run. |
|  | If we want the loop to break based on a condition other than the number of times it runs, we should use a while loop.  8.  Bubble   |  | | --- | | package com.assignment; | |  |  | |  | public class q8BubbleSort { | |  | void bubbleSort(int arr[]) | |  | { | |  | int n = arr.length; | |  | for (int i = 0; i < n-1; i++) | |  | for (int j = 0; j < n-i-1; j++) | |  | if (arr[j] > arr[j+1]) | |  | { | |  | // swap arr[j+1] and arr[j] | |  | int temp = arr[j]; | |  | arr[j] = arr[j+1]; | |  | arr[j+1] = temp; | |  | } | |  | } | |  |  | |  | /\* Prints the array \*/ | |  | void printArray(int arr[]) | |  | { | |  | int n = arr.length; | |  | for (int i=0; i<n; ++i) | |  | System.out.print(arr[i] + " "); | |  | System.out.println(); | |  | } | |  |  | |  | // Driver method to test above | |  | public static void main(String args[]) | |  | { | |  | q8BubbleSort ob = new q8BubbleSort(); | |  | int arr[] = {64, 34, 25, 12, 22, 11, 90}; | |  | ob.bubbleSort(arr); | |  | System.out.println("Sorted array"); | |  | ob.printArray(arr); | |  | } | |  | } |   Heap   |  | | --- | |  | |  |  | |  | public class q8HeapSort { | |  | public void sort(int arr[]) | |  | { | |  | int n = arr.length; | |  |  | |  | // Build heap (rearrange array) | |  | for (int i = n / 2 - 1; i >= 0; i--) | |  | heapify(arr, n, i); | |  |  | |  | // One by one extract an element from heap | |  | for (int i=n-1; i>=0; i--) | |  | { | |  | // Move current root to end | |  | int temp = arr[0]; | |  | arr[0] = arr[i]; | |  | arr[i] = temp; | |  |  | |  | // call max heapify on the reduced heap | |  | heapify(arr, i, 0); | |  | } | |  | } | |  |  | |  | // To heapify a subtree rooted with node i which is | |  | // an index in arr[]. n is size of heap | |  | void heapify(int arr[], int n, int i) | |  | { | |  | int largest = i; // Initialize largest as root | |  | int l = 2\*i + 1; // left = 2\*i + 1 | |  | int r = 2\*i + 2; // right = 2\*i + 2 | |  |  | |  | // If left child is larger than root | |  | if (l < n && arr[l] > arr[largest]) | |  | largest = l; | |  |  | |  | // If right child is larger than largest so far | |  | if (r < n && arr[r] > arr[largest]) | |  | largest = r; | |  |  | |  | // If largest is not root | |  | if (largest != i) | |  | { | |  | int swap = arr[i]; | |  | arr[i] = arr[largest]; | |  | arr[largest] = swap; | |  |  | |  | // Recursively heapify the affected sub-tree | |  | heapify(arr, n, largest); | |  | } | |  | } | |  |  | |  | /\* A utility function to print array of size n \*/ | |  | static void printArray(int arr[]) | |  | { | |  | int n = arr.length; | |  | for (int i=0; i<n; ++i) | |  | System.out.print(arr[i]+" "); | |  | System.out.println(); | |  | } | |  |  | |  | // Driver program | |  | public static void main(String args[]) | |  | { | |  | int arr[] = {12, 11, 13, 5, 6, 7}; | |  | int n = arr.length; | |  |  | |  | q8HeapSort ob = new q8HeapSort(); | |  | ob.sort(arr); | |  |  | |  | System.out.println("Sorted array is"); | |  | printArray(arr); | |  | } | |  | } |   9.   |  | | --- | |  | |  |  | |  | public class q9 { | |  | public static void main(String[] args) { | |  |  | |  | // year to be checked | |  | int year = 2000; | |  | boolean leap = false; | |  |  | |  | // if the year is divided by 4 | |  | if (year % 4 == 0) { | |  |  | |  | // if the year is century | |  | if (year % 100 == 0) { | |  |  | |  | // if year is divided by 400 | |  | // then it is a leap year | |  | if (year % 400 == 0) | |  | leap = true; | |  | else | |  | leap = false; | |  | } | |  |  | |  | // if the year is not century | |  | else | |  | leap = true; | |  | } | |  |  | |  | else | |  | leap = false; | |  |  | |  | if (leap) | |  | System.out.println(year + " is a leap year."); | |  | else | |  | System.out.println(year + " is not a leap year."); | |  | } | |  | } |   10.   |  | | --- | |  | |  | import java.util.Scanner; | |  | public class q10 { | |  | public static void main(String args[]) | |  | { | |  | Scanner sc = new Scanner(System.in); | |  | System.out.println("Enter the two numbers: "); | |  | int x = sc.nextInt(); | |  | int y = sc.nextInt(); | |  | System.out.println("The GCD of two numbers is: " + findGCD(x,y)); | |  | } | |  | static int findGCD(int x, int y) | |  | { | |  | int r=0, a, b; | |  | a = (x > y) ? x : y; // a is greater number | |  | b = (x < y) ? x : y; // b is smaller number | |  | r = b; | |  | while(a % b != 0) | |  | { | |  | r = a % b; | |  | a = b; | |  | b = r; | |  | } | |  | return r; | |  | } |   11.   |  | | --- | | ; | |  | import java.util.\*; | |  | public class q11 { | |  | public void convertBinary(int num){ | |  | int binary[] = new int[40]; | |  | int index = 0; | |  | while(num > 0){ | |  | binary[index++] = num%2; | |  | num = num/2; | |  | } | |  | for(int i = index-1;i >= 0;i--){ | |  | System.out.print(binary[i]); | |  | } | |  | } | |  |  | |  | public static void main(String[] args) { | |  | q11 obj=new q11(); | |  | Scanner in = new Scanner(System.in); | |  | System.out.println("Enter decimal number: "); | |  | int num = in.nextInt(); | |  | System.out.print("\nBinary representation is:"); | |  | obj.convertBinary(num); | |  | } | |  | } |   12.   |  | | --- | |  | |  |  | |  | public class q12 { | |  | public static void main(String[] args) { | |  |  | |  | int length = 5; | |  | int height = 5; | |  |  | |  | for (int i = 0; i <= height; i++) | |  | { | |  | if (i % 2 == 0) | |  | { | |  | System.out.print("\* "); | |  | } | |  | else | |  | { | |  | System.out.print(" "); | |  | } | |  | for (int j = 0; j <= length; j++) | |  | { | |  | if (j % 2 == 0) | |  | { | |  | System.out.print(" "); | |  | } | |  | else if (j != length || i % 2 != 0) | |  | { | |  | System.out.print("\* "); | |  | } | |  | } | |  | System.out.println(""); | |  | } | |  | } | |  | } | |