**Occurrence of character**

private static void characterCount(String inputString)

    {

 //Creating a HashMap containing char as a key and occurrences as a valur

HashMap<Character, Integer> charCountMap = new HashMap<Character, Integer>();

        //Converting given string to char array

        char[] strArray = inputString.toCharArray();

        //checking each char of strArray

        for (char c : strArray)

        {

            if(charCountMap.containsKey(c))

            {

     //If char 'c' is present in charCountMap, incrementing it's count by 1

              charCountMap.put(c, charCountMap.get(c)+1);

            }

            else

            {

     //If char 'c' is not present in charCountMap//putting 'c' into charCountMap with 1 as it's value

                charCountMap.put(c, 1);

            }

        }

        //Printing inputString and charCountMap

        System.out.println(inputString+" : "+charCountMap);

    }

    public static void main(String[] args)

    {

       characterCount("Java J2EE Java JSP J2EE");

       characterCount("All Is Well");

       characterCount("Done And Gone");

    }

}

**Palindrome of string**

 static boolean isPalindrome(String str)

    {

 // Pointers pointing to the beginning

        // and the end of the string

        int i = 0, j = str.length() - 1;

        // While there are characters toc compare

        while (i < j) {

 // If there is a mismatch

   if (str.charAt(i) != str.charAt(j))

                return false;

            i++;

            j--;

        }

        // Given string is a palindrome

        return true;

    }

*/\*****\* Java method to check if number is palindrome or not*** *\*/*  
    **public** **static** **boolean** isPalindrome(**int** number) {  
        **int** palindrome = number; *// copied number into variable*  
        **int** reverse = 0;  
  
        while (palindrome != 0) {  
            **int** remainder = palindrome % 10;  
            reverse = reverse \* 10 + remainder;  
            palindrome = palindrome / 10;  
        }  
  
        *// if original and reverse of number is equal means*  
        *// number is palindrome in Java*  
        if (number == reverse) {  
            **return** **true**;  
        }  
        **return** **false**;  
    }

1. **int** length = original.length();
2. **for** ( **int** i = length - 1; i >= 0; i-- )
3. reverse = reverse + original.charAt(i);
4. **if** (original.equals(reverse))
5. System.out.println("Entered string/number is a palindrome.");
6. **else**
7. System.out.println("Entered string/number isn't a palindrome.");

**// creating ArrayList with duplicate elements**

List<Integer> primes = new ArrayList<Integer>();

primes.add(2);

primes.add(3);

primes.add(5);

primes.add(7); //duplicate

primes.add(7);

primes.add(11);

// let's print arraylist with duplicate

System.out.println("list of prime numbers : " + primes);

Set<Integer> primesWithoutDuplicates = new LinkedHashSet<Integer>(primes);

// now let's clear the ArrayList so that we can copy all elements from LinkedHashSet

primes.clear();

// copying elements but without any duplicates

primes.addAll(primesWithoutDuplicates);

System.out.println("list of primes without duplicates : " + primes);

}

* list all name who receding at US

//table[@id='ctl00\_MainContent\_orderGrid']//td[text()='US']//preceding-sibling::td[4]

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Street** | **City** | **State** |
| Paul | 5, valley | Las Vegas | US |
| Mark | 9, Maple Valley | Whitestone | Canada |
| Steve | 17, Park Avenue | Salmon | Canada |
| Charles | 8 Park | Bring tone | US |
| Bob | 2, avenue | Las Vegas | US |