**Assignment . 7**

**Problem Statement:**

Implement a generic program using any collection class to count the number of elements in a collection that have a specific property such as even numbers, odd number, prime number and palindromes.

**Objectives:** To learn the concept templates and generic programming

**Theory:**

* 1. Java Generic methods
     + Syntax to declare class
     + Instance variable in Java
     + Method in Java
     + ‘new’ keyword in Java
  2. Generic classes
     + - multiple Type parameters

1. Advantages of Generics:

Generic Types

* Generic type represents classes, interfaces and methods in a type safe manner
* Generic types can act on any type of data
* All Generic types are subclasses of Object class, it acts on Objects only
* Generic types act on advanced data type only
* It is not possible to create an object to Generic type itself
* Using generic types, we can avoid casting in many cases Generic Class:

When we create a class with an instance variable to store an Integer object, it can be used to store Integer type data only

We cannot use that instance variable to store a Float class object or a String type Object To store different types of data into a class, we have to write the same class again and again by changing the data type of the variables This can be avoided using a generic class A generic class represents a class that is type-safe This means a generic class can act upon any data type Generic classes and generic interfaces are also called „parameterized types‟ because they use a parameter that determines which data type they should work upon

**Generic Method:** We can make a method alone as generic method by writing the generic

parameter before the method return type as:

returntypemethodname ()

{

Method code;

}

eg: void display\_data () { Method body; }

**Generic Interface**:

It is possible to develop an interface using generic type concept. The general form of generic interface looks like:

interface interface\_name

{ //method that accepts any object return\_typemethod\_name ( T object\_name ); } Here, T represents any data type which is used in the interface.

We can write an implementation class for the above interface as: class class\_name implements

interface\_name

{ public return\_typemethod\_name ( T object\_name ) { //provide body of the method }

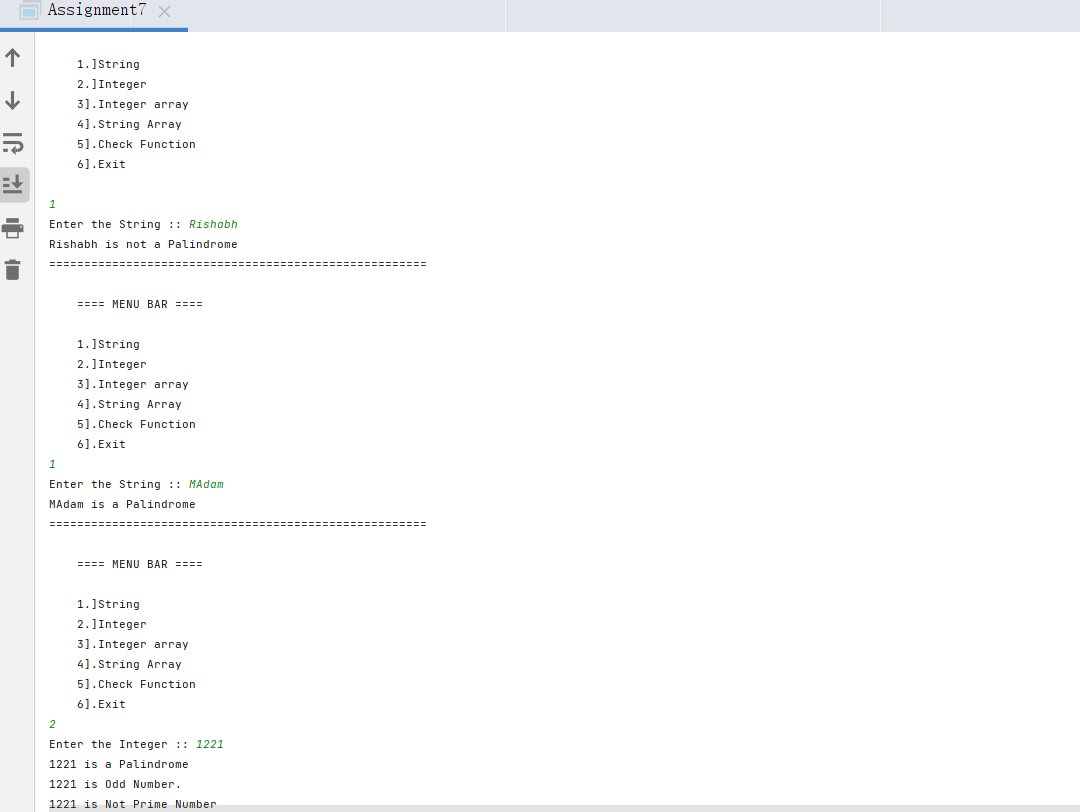
}

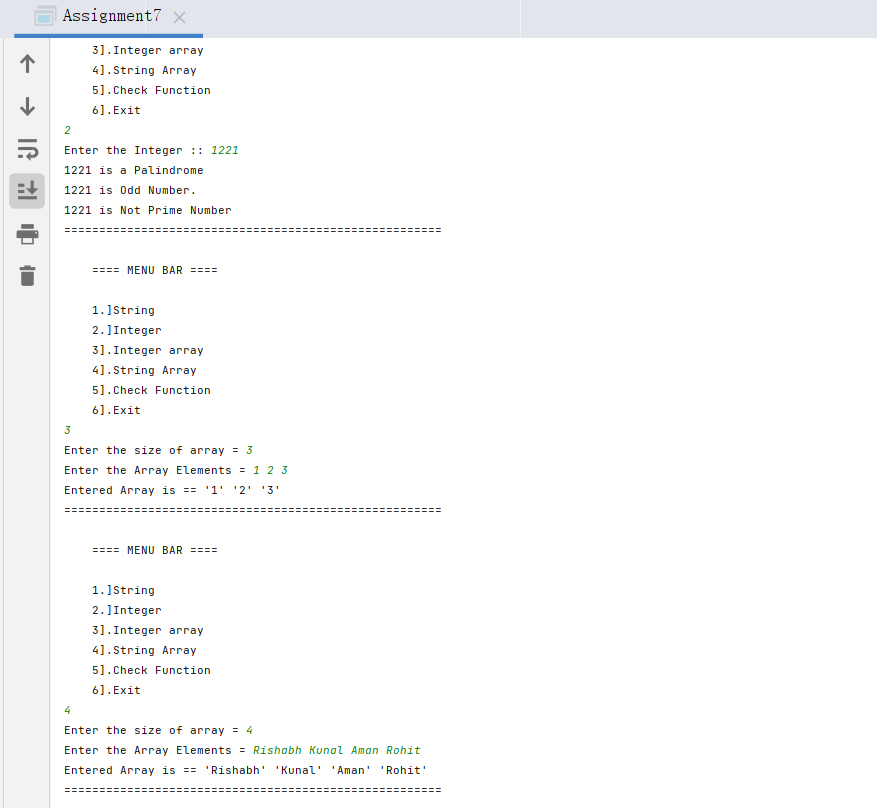
**Algorithm:**

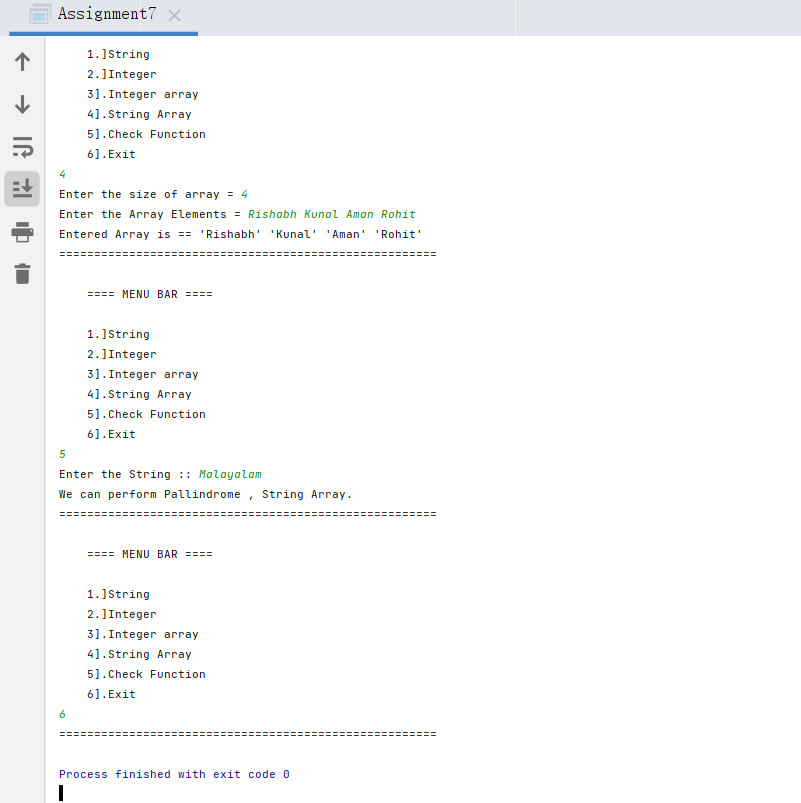
* 1. CREATE A NUMBER CLASS AND FOR ARRAY ELEMENTS TAKE THE INPUT FROM THE USER DISPLAY IT IN GENERIC METHOD
  2. CREATE A PALLINDROME CLASS AND THEN AND PERFORM PALLINDROME FUNCTION ON IT IN GENERIC METHOD
  3. CREATE A PRIME CLASS AND THEN AND PERFORM PRIME FUNCTION ON IT IN GENERIC METHOD
  4. CREATE A MAIN CLASS AND THEN CALL ALL THE FUNCTIONS BY SWITCH CASE STATEMENT.

**Code**

package com.company;  
  
import java.util.Objects;  
import java.util.Scanner;  
  
 class Number{  
 public static < E > void arr( ) {  
 Scanner sc =new Scanner(System.*in*);  
 System.*out*.print("Enter the size of array = ");  
 int n=sc.nextInt();  
  
 Object[] arr=new Object[n];  
  
 System.*out*.print("Enter the Array Elements = ");  
 for(int i =0; i<n ;i++) {  
 arr[i]=sc.next();  
 }  
  
 System.*out*.print("Entered Array is == ");  
 for(Object element : arr) {  
 System.*out*.printf("'%s' ", element);  
 }  
  
 System.*out*.println();  
 }  
  
public static < T > void Pallindrome(T s){  
 String s1=(String)s;  
  
 s1=s1.toLowerCase();  
  
 StringBuffer sb = new StringBuffer(s1);*//doubt* String ss= new String(sb.reverse());  
  
 if(Objects.*equals*(s1, ss))  
 System.*out*.println(s+" is a Palindrome");  
 else  
 System.*out*.println(s+" is not a Palindrome");  
}  
  
public static <T>void evenodd(T a){  
 if((int)a%2==0)  
 System.*out*.println(a+" is an Even Number.");  
  
 else  
 System.*out*.println(a+" is Odd Number.");  
}  
  
public static <T>void prime(T a) {  
 if ((int) a == 1)  
 System.*out*.println(a + " is Not Prime NNumber");  
  
 else if ((int) a == 2)  
 System.*out*.println(a + " is Prime Number");  
  
 else if ((int) a % 2 == 0 && (int) a > 2)  
 System.*out*.println(a + " is Not Prime Number");  
  
 else {  
 double b = Math.*sqrt*((int) a) + 1;  
 int temp = 0;  
 for (int i = 3; i < b; i = i + 2) {  
 if ((int) a % i == 0)  
 temp = 1;  
 }  
 if (temp == 1)  
 System.*out*.println(a + " is Not Prime Number");  
 else  
 System.*out*.println(a + " is Prime NNumber");  
 }  
}  
  
 public static <T>void checkfun(T s) {  
  
 try {  
  
 int b = Integer.*parseInt*((String) s);  
 System.*out*.println("We can perform Palindrome , Integer Array , Check Prime , EvenOdd Function.");  
 }  
  
 catch (NumberFormatException e) {  
 System.*out*.println("We can perform Pallindrome , String Array.");  
 }  
 }  
 }  
  
public class Assignment7 {  
 public static void main(String[] args) {  
 String s;  
 Scanner sc =new Scanner(System.*in*);  
  
 aa:  
 while(true) {  
 System.*out*.println("\n\t==== MENU BAR ====\n\n\t1.]String \n\t2.]Integer"  
 + "\n\t3].Integer array \n\t4].String Array"  
 + "\n\t5].Check Function\n\t6].Exit");  
  
 int c =sc.nextInt();  
  
  
 switch(c) {  
  
 case 1:  
 System.*out*.print("Enter the String :: ");  
 s =sc.next();  
 Number.*Pallindrome*(s);  
 System.*out*.println("======================================================");  
 break;  
  
 case 2:  
 System.*out*.print("Enter the Integer :: ");  
 s =sc.next();  
 Number.*Pallindrome*(s);  
 Number.*evenodd*(Integer.*parseInt*(s));  
 Number.*prime*(Integer.*parseInt*(s));  
 System.*out*.println("======================================================");  
 break;  
  
  
 case 3: *//if input is 3* case 4: *//if input is 4* Number.*arr*();  
 System.*out*.println("======================================================");  
 break;  
  
 case 5: *//if input is 5* System.*out*.print("Enter the String :: ");  
 String ss =sc.next();  
 Number.*checkfun*(ss);  
 System.*out*.println("======================================================");  
 break;  
  
 case 6: *//if input is 6* System.*out*.println("======================================================");  
 break aa;  
  
 default: *//default Statement* System.*out*.println("Invalid Input !!!");  
 System.*out*.println("======================================================");  
  
 }  
 }  
 }  
}







**Conclusion:**

**The concept of generics in OOP is understood with this Assignment**

**And the assignment is successfully completed.**