

**Ex No: 01 WRITE A JAVA PROGRAM TO CHECK WHETHER THE GIVEN STRING IS
Date: PALINDROME OR NOT.**

AIM:

To write a JAVA program to check whether the given string is palindrome is not.

ALGORITHM:

1. Define a string.
2. Define a variable flag and set it to true.
3. Convert the string to lowercase to make the comparison case-insensitive.
4. Now, iterate through the string forward and backward, compare one character at a time till middle of the string is reached.
5. If any of the character doesn't match, set the flag to false and break the loop.
6. At the end of the loop, if flag is true, it signifies string is a palindrome.
7. If flag is false, then string is not a palindrome.

PROGRAM:

```
public class Palindrome
{
public static void main(String[] args) {
String string = "Kayak";
boolean flag = true;

//Converts the given string into lowercase
string = string.toLowerCase();

//Iterate the string forward and backward, compare one character at a time
//till middle of the string is reached
for(int i = 0; i < string.length()/2; i++){
if(string.charAt(i) != string.charAt(string.length()-i-1)){
flag = false;
break;
}
}
```

```
if(flag)
System.out.println("Given string is palindrome");
else
System.out.println("Given string is not a palindrome");
}
}
```

OUTPUT

Given string is palindrome

RESULT:

Thus the program will be executed successfully.

Ex no:2 DEVELOP A JAVA APPLICATION TO GENERATE ELECTRICITY BILL

Date:

Aim:

To develop a JAVA application to generate the electricity bill.

ALGORITHM:

- 1.Start the program.
- 2.Define the constant UNIT_RATE for the rate per unit consumed and TAX_RATE for the tax rate.
- 3.Define the function calculateBill that takes the number of units consumed as an argument and returns the bill amount.
- 4.Declare variables billAmount and taxAmount.
- 5.Calculate the bill amount by multiplying the units consumed by UNIT_RATE.
- 6.Calculate the tax amount by multiplying the bill amount by TAX_RATE.
- 7.Add the tax amount to the bill amount.
- 8.Return the bill amount.
- 9.Define the main FunctionFunction.
- 10.Declare variables units and totalBill.
- 11.Prompt the user to input the number of units consumed.
- 12.Read the input and store it in the units variable.
- 13.Call the calculateBill FunctionFunction, passing the units variable as an argument, and store the
- 14.returned value in totalBill.
- 15.Display the bill amount by printing "Electricity Bill Amount: \$totalBill".
- 16.End the program.

PROGRAM:

```
import java.util.*;
class ComputeElectricityBill
{
public static void main(String args[])
{
int units=280;

double billpay=0;

if(units<100)
```

```
{  
billpay=units*1.20;  
}  
else if(units<300)  
{  
billpay=100*1.20+(units-100)*2;  
}  
else if(units>300)  
{  
billpay=100*1.20+200*2+(units-300)*3;  
}  
  
System.out.println("Bill to pay : " + billpay);  
}  
}
```

OUTPUT:

Bill to pay : 480.0

RESULT:

Thus the program will be executed successfully.

Ex. No : 03 DEVELOP A JAVA APPLICATION IMPLEMENT CURRENCY CONVERTER
Date : USING PACKAGES.

AIM:

To develop a JAVA application implement currency converter using packages.

ALGORITHM:

1. Take a floating input from the user into a variable that represents Indian rupees count.
2. Assign the current conversion rate into another variable. (You can take it as input from the user too)
3. Now, write a mathematical statement where variable that represents amount of Indian currency divides with conversion rate and store the computed value into a variable that represents dollar amount.
As in;
$$4. \text{Dollar Amount} = \text{Indian Amount} / \text{Conversion Rate}$$
5. Print the dollar amount.

PROGRAM:

```
CurrencyC.java
package cc;
import java.util.*;
public class CurrencyC {
double inr, usd;
double euro, yen;
Scanner in = new Scanner(System.in);
public void dollartorupee() {
System.out.println("Enter dollars to convert into Rupees: ");
usd = in.nextInt();
inr = usd * 81.83;
System.out.println("Dollar =" + usd + " equal to INR =" + inr);
System.out.println("\n ");
}
public void rupeetodollar() {
System.out.println("Enter Rupee to convert into Dollars:");
inr = in.nextInt();
usd = inr / 81.83;
System.out.println("Rupee =" + inr + "equal to Dollars=" + usd);
}
```

```

public void eurotorupee() {
    System.out.println("Enter Euro to convert into Rupees:");
    euro = in.nextInt();
    inr = euro * 79.06;
    System.out.println("Euro =" + euro + " equal to INR=" + inr);
    System.out.println("\n");
}
public void rupeeetoeuro() {
    System.out.println("Enter Rupeesto convert into Euro:");
    inr = in.nextInt();
    euro = (inr / 79.06);
    System.out.println("Rupee =" + inr + "equal to Euro=" + euro);
    System.out.println("\n");
}
public void yentoruppe() {
    System.out.println("Enter Yen to convert into Rupees:");
    yen = in.nextInt();
    inr = yen * 0.57;
    System.out.println("Yen =" + yen + " equal to INR=" + inr);
    System.out.println("\n");
}
public void ruppetoyen() {
    System.out.println("Enter Rupeesto convert into Yen:");
    inr = in.nextInt();
    yen = (inr / 0.57);
    System.out.println("INR=" + inr + "equal to YEN" + yen);
    System.out.println("\n");
}
}
}
}
}

```

OUTPUT:

```

Enter dollars to convert into Rupees:1
Dollar =1.0 equal to INR=81.83
Enter Rupee to convert into Dollars: 80
Rupee =80.0equal to Dollars=0.977636563607479

```

RESULT:

Thus the program currency converter will be executed successfully.

Ex. No :04

DESIGN A JAVA INTERFACE FOR ADT STACK.IMPLEMENT THIS INTERFACE

Date :

USING ARRAY.

AIM:

To design a JAVA interface for ADT stack and to implement this interface using array.

ALGORITHM:

- 1.Create the interface stack operation with method declarations for push and pop.
- 2.Create the class a stack which implements the interface and provides implementation for the methods push and pop. Also define the method for displaying the values stored in the stack. Handle the stack overflow and stack underflow condition .
- 3.Create the class test stack .Get the choice from the user for the operation to be performed . and also handle the exception that occur while performing the stack operation.
- 4.Create the object and invoke the method for push, pop, display based on the input from the user.

PROGRAM:

```
// Java Program to Implement Stack in Java Using Array and  
// Generics
```

```
// Importing input output classes
```

```
import java.io.*;
```

```
// Importing all utility classes
```

```
import java.util.*;
```

```
// user defined class for generic stack
```

```
class stack<T> {
```

```
// Empty array list
```

```
ArrayList<T> A;
```

```
// Default value of top variable when stack is empty
```

```
int top = -1;
```

```

// Variable to store size of array
int size;

// Constructor of this class
// To initialize stack
stack(int size)
{
// Storing the value of size into global variable
this.size = size;

// Creating array of Size = size
this.A = new ArrayList<T>(size);
}

// Method 1
// To push generic element into stack
void push(T X)
{
// Checking if array is full
if (top + 1 == size) {

// Display message when array is full
System.out.println("Stack Overflow");
}
else {

// Increment top to go to next position
top = top + 1;

// Over-writing existing element
if (A.size() > top)
A.set(top, X);

else

```



```

// Creating new element
A.add(X);
}
}
// Method 2
// To return topmost element of stack
T top()
{
// If stack is empty
if (top == -1) {

// Display message when there are no elements in
// the stack
System.out.println("Stack Underflow");

return null;
}

// else elements are present so
// return the topmost element
else
return A.get(top);
}

// Method 3
// To delete last element of stack
void pop()
{
// If stack is empty
if (top == -1) {

// Display message when there are no elements in
// the stack
System.out.println("Stack Underflow");
}
}

```

else

```
// Delete the last element
// by decrementing the top
top--;
}
```

```
// Method 4
// To check if stack is empty or not
boolean empty() { return top == -1; }
```

```
// Method 5
// To print the stack
// @Override
public String toString()
{
```

```
String Ans = "";
```

```
for (int i = 0; i < top; i++) {
Ans += String.valueOf(A.get(i)) + "->";
}
```

```
Ans += String.valueOf(A.get(top));
```

```
return Ans;
}
}
```

```
// Main Class
```

```
public class GFG {
```

```
// main driver method
```

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

```

// Integer Stack

// Creating an object of Stack class
// Declaring objects of Integer type
stack<Integer> s1 = new stack<>(3);

// Pushing elements to integer stack - s1

// Element 1 - 10
s1.push(10);
// Element 2 - 20
s1.push(20);
// Element 3 - 30
s1.push(30);

// Print the stack elements after pushing the
// elements
System.out.println(
"s1 after pushing 10, 20 and 30 :\n" + s1);

// Now, pop from stack s1
s1.pop();

// Print the stack elements after popping few
// element/s
System.out.println("s1 after pop :\n" + s1);

// String Stack

// Creating an object of Stack class
// Declaring objects of Integer type
stack<String> s2 = new stack<>(3);

// Pushing elements to string stack - s2

```

```

// Element 1 - hello
s2.push("hello");
// Element 2 - world
s2.push("world");
// Element 3 - java
s2.push("java");

// Print string stack after pushing above string
// elements
System.out.println(
"\ns2 after pushing 3 elements :\n" + s2);

System.out.println(
"s2 after pushing 4th element :");

// Pushing another element to above stack

// Element 4 - GFG
s2.push("GFG");

// Float stack

// Creating an object of Stack class
// Declaring objects of Integer type
stack<Float> s3 = new stack<>(2);

// Pushing elements to float stack - s3

// Element 1 - 100.0
s3.push(100.0f);
// Element 2 - 200.0
s3.push(200.0f);

// Print string stack after pushing above float

```

```
// elements
System.out.println(
"\ns3 after pushing 2 elements :\n" + s3);

// Print and display top element of stack s3
System.out.println("top element of s3:\n"
+ s3.top());
}
}
```

OTUPUT:

s1 after pushing 10, 20 and 30 :

10->20->30

s1 after pop :

10->20

s2 after pushing 3 elements :

hello->world->java

s2 after pushing 4th element :

Stack Overflow

s3 after pushing 2 elements :

100.0->200.0

top element of s3:

200.0

RESULT:

Thus the java application for stack operations has been implemented and executed successfully.

Ex.No.:05

WRITE A PROGRAM TO PERFORM STRING OPERATIONS USING ARRAY

Date :

LIST.

AIM:

To write a program to perform string operations using array list.

ALGORITHM:

1. Create the class array list example. Create the object for the arraylist class.
2. Display the options to the user for performing string handling .
3. Use the function add() to append the string at the end and to insert the string at the particular index.
4. The function sort () is used to sort the elements in the array list.
5. The function index of() is used to search whether the element is in the array list or not.
6. The function starts With () is used to find whether the element starts with the specified character.
7. The function remove() is used to remove the element from the array list.
8. The function size() is used to determine the number of elements in the array list.

PROGRAM:

```
import java.util.*;

import java.io.*;

public class arraylistexample

{

public static void main(String args[])throws IOException

{

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

DataInputStream in=new DataInputStream(System.in);

int c,ch;

int i,j;

String str,str1;
```

```

do

{

System.out.println(" 1. Append at end \t 2.Insert at particular index \t 3.Search \t");

System.out.println( "4.List string that starting with letter \t");

System.out.println("5.Size \t 6.Remove \t 7.Sort\t 8.Display\t" );

System.out.println("Enter the choice ");

c=Integer.parseInt(in.readLine());

switch(c)

{

case 1:

{

System.out.println("Enter the string ");

str=in.readLine();

obj.add(str);

break;

}

case 2:

{

System.out.println("Enter the string ");

str=in.readLine();

System.out.println("Specify the index/position to insert");

i=Integer.parseInt(in.readLine());

```

```
obj.add(i-1,str);

System.out.println("The array list has following elements:"+obj);

break;

}

case 3:

{

System.out.println("Enter the string to search ");

str=in.readLine();

j=obj.indexOf(str);

if(j==-1)

System.out.println("Element not found");

else

System.out.println("Index of:"+str+"is"+j);

break;

}

case 4:

{

System.out.println("Enter the character to List string that starts with specified character");

str=in.readLine();

for(i=0;i<(obj.size()-1);i++)

{

str1=obj.get(i);
```



```
if(str1.startsWith(str))

{

System.out.println(str1);

}

}

break;

}

case 5:

{

System.out.println("Size of the list "+obj.size());

break;

}

case 6:

{

System.out.println("Enter the element to remove");

str=in.readLine();

if(obj.remove(str))

{

System.out.println("Element Removed"+str);

}

else

{
```

```

        System.out.println("Element not present");
    }

    break;
}

case 7:

{
    Collections.sort(obj);

    System.out.println("The array list has following elements:"+obj);

    break;
}

case 8:

{
    System.out.println("The array list has following elements:"+obj);

    break;
}

}

System.out.println("enter 0 to break and 1 to continue");

ch=Integer.parseInt(in.readLine());

}while(ch==1);

}

}

```

OUTPUT:

1. Append at end
 2. Insert at particular index
 3. Search
 4. List string that starting with letter
 5. Size
 6. Remove
 7. Sort
 8. Display
- Enter the choice
Enter the string
enter 0 to break and 1 to continue

RESULT:

Thus the java program to perform string operations using ArrayList has been implemented and executed successfully.

Ex.No.: 06

WRITE A JAVA PROGRAM TO IMPLEMENT USER DEFINED EXCEPTION

Date : HANDLING

AIM:

To write a JAVA program to implement user defined exception handling.

ALGORITHM:

- 1.The test Exception() method throws exceptions by using the throw keyword. In order to notify the caller of the potential exception kinds, the throws keyword is included in the method signature.
- 2.The try-catch block is used in the main() method to handle exceptions.
- 3.The finally block is then completed, and the test Exception(-20) is never executed as a result of the exception.
- 4.The print Stack Trace() function of the Exception class is helpful for debugging.

PROGRAM:

// A Class that represents use-defined exception

```
class MyException extends Exception {  
    public MyException(String s)  
    {  
        // Call constructor of parent Exception  
        super(s);  
    }  
}
```

```
// A Class that uses above MyException  
public class Main {  
    // Driver Program  
    public static void main(String args[])  
    {  
        try {  
            // Throw an object of user defined exception  
            throw new MyException("GeeksGeeks");  
        }  
        catch (MyException ex) {  
            System.out.println("Caught");  
        }  
    }  
}
```

```
// Print the message from MyException object
System.out.println(ex.getMessage());
}
}
}
```

OUTPUT:

Caught
Geeks Geeks

RESULT:

Thus the program will be executed successfully.

Ex.No.:07

WRITE A JAVA PROGRAM THAT IMPLEMENTS A MULTI THREADED APPLICATION

AIM:

To write a java program that implements a multi threaded application.

ALGORITHM:

1.Multi-threading is a programming concept in which the application can create a small unit of tasks to execute in parallel.

2. A simple program runs in sequence and the code statements execute one by one.

3.The programming language supports creating multiple threads and passes them to the operating system to run in parallel, it's called multi-threading.

PROGRAM:

```
class RunnableDemo implements Runnable {  
  
    private Thread t;  
  
    private String threadName;  
  
    RunnableDemo(String name) {  
  
        threadName = name;  
  
        System.out.println("Creating " + threadName);  
  
    }  
  
    public void run() {  
  
        System.out.println("Running " + threadName);
```

```

try {

    for (int i = 4; i > 0; i--) {

        System.out.println("Thread: " + threadName + ", " + i);

        // Let the thread sleep for a while.

        Thread.sleep(50);

    }

} catch (InterruptedException e) {

    System.out.println("Thread " + threadName + " interrupted.");

}

System.out.println("Thread " + threadName + " exiting.");

}

public void start() {

    System.out.println("Starting " + threadName);

    if (t == null) {

        t = new Thread(this, threadName);

        t.start();

    }

}

}

public class TestThread {

```

```
public static void main(String args[]) {  
  
    RunnableDemo R1 = new RunnableDemo("Thread-1");  
  
    R1.start();  
  
    RunnableDemo R2 = new RunnableDemo("Thread-2");  
  
    R2.start();  
  
}  
  
}
```

OUTPUT:

```
Creating Thread-1  
Starting Thread-1  
Creating Thread-2  
Starting Thread-2  
Running Thread-1  
Running Thread-2  
Thread: Thread-1, 4  
Thread: Thread-2, 4  
Thread: Thread-1, 3  
Thread: Thread-2, 3  
Thread: Thread-1, 2  
Thread: Thread-2, 2  
Thread: Thread-1, 1  
Thread: Thread-2, 1  
Thread Thread-1 exiting.  
Thread Thread-2 exiting.
```

RESULT:

Thus the program multi threaded application will be executed successfully.

**Ex.No:08 WRITE A JAVA PROGRAM TO FIND THE MAXIMUM VALUE FROM THE GIVEN TYPE
Date: OF ELEMENTS USING A GENERIC FUNCTION.**

AIM:

To write a java program to find the maximum value from the given type of elements using a generic function.

ALGORITHM:

1. Create a class Myclass to implement generic class and generic methods.
2. Get the set of the values belonging to specific data type.
3. Create the objects of the class to hold integer, character and double values.
4. Create the method to compare the values and find the maximum value stored in the array.
5. Invoke the method with integer, character or double values . The output will be displayed based on the data type passed to the method.

PROGRAM:

```
class MyClass<T extends Comparable<T>>

{

T[] vals;

MyClass(T[] o)

{

vals = o;

}

public T min()

{

T v = vals[0];
```

```
for(int i=1; i < vals.length; i++)
```

```
if(vals[i].compareTo(v) < 0)
```

```
v = vals[i];
```

```
return v;
```

```
}
```

```
public T max()
```

```
{
```

```
T v = vals[0];
```

```
for(int i=1; i < vals.length;i++)
```

```
if(vals[i].compareTo(v) > 0)
```

```
v = vals[i];
```

```
return v;
```

```
}
```

```
}
```

```
class gendemo
```

```
{
```

```
public static void main(String args[])
```

```
{
```

```
int i;
```

```
Integer inums[]={10,2,5,4,6,1};
```

```
Character chs[]={'v','p','s','a','n','h'};
```

```
Double d[]={20.2,45.4,71.6,88.3,54.6,10.4};

MyClass<Integer> iob = new MyClass<Integer>(inums);

MyClass<Character> cob = new MyClass<Character>(chs);

MyClass<Double>dob = new MyClass<Double>(d);

System.out.println("Max value in inums: " + iob.max());

System.out.println("Min value in inums: " + iob.min());

System.out.println("Max value in chs: " + cob.max());

System.out.println("Min value in chs: " + cob.min());

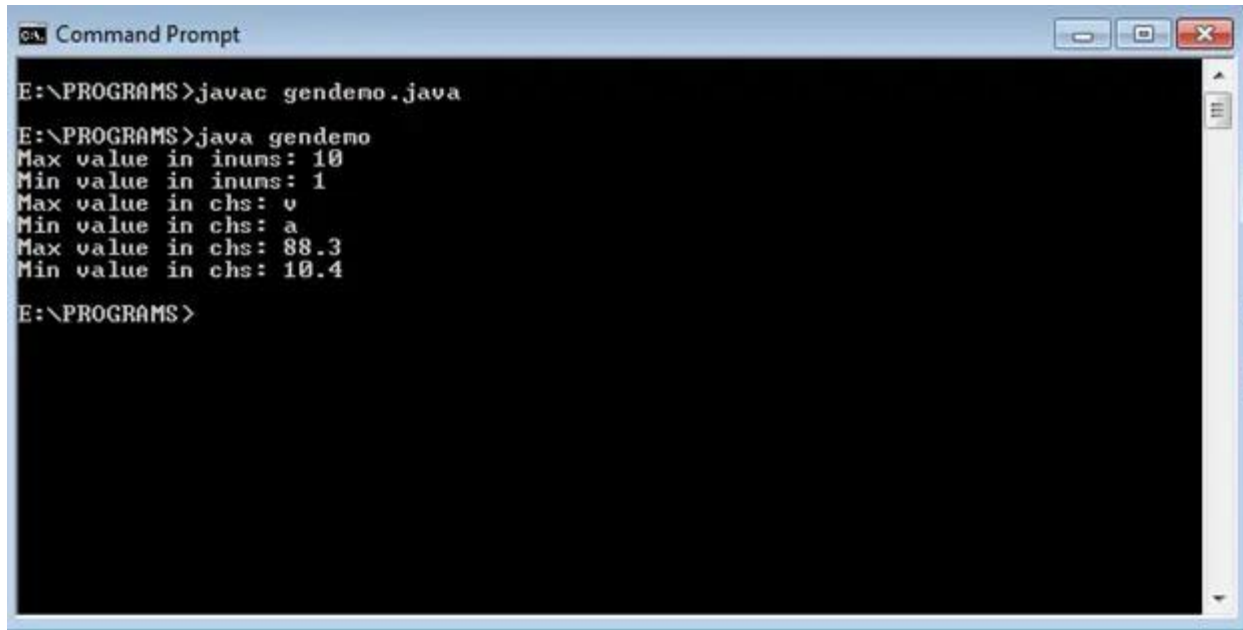
System.out.println("Max value in chs: " + dob.max());

System.out.println("Min value in chs: " + dob.min());

}

}
```

OUTPUT:



The image shows a Windows Command Prompt window titled "Command Prompt". The window has a standard Windows interface with a title bar, minimize, maximize, and close buttons. The background is black, and the text is white. The command prompt shows the following sequence of commands and output:

```
E:\PROGRAMS>javac gendemo.java
E:\PROGRAMS>java gendemo
Max value in inums: 10
Min value in inums: 1
Max value in chs: v
Min value in chs: a
Max value in chs: 88.3
Min value in chs: 10.4
E:\PROGRAMS>
```

RESULT:

Thus a java program to find the maximum value from the given type of elements has been implemented using generics and executed successfully.

