Ex. No.:1

ELECTRICITY BILL GENERATION

Aim:

To develop a Java application to generate Electricity bill.

Procedure:

1. Start the program
2. Create consumer class with the following members: Consumer no., consumer name, previous month reading, current month reading, type of EB connection (i.e domestic or commercial).

3.If the type of the EB connection is domestic, calculate the amount to be paid as follows:

- First 100 units - Rs.1 per unit
- 101-200 units - Rs.2.50 per unit
- 201 -500 units - Rs. 4 per unit
- > 501 units - Rs.6 per unit

4. If the type of the EB connection is commercial, calculate the amount to be paid as follows:

- First 100 units - Rs. 2 per unit
- 101-200 units - Rs. 4.50 per unit
- 201 -500 units - Rs. 6 per unit
- > 501 units - Rs. 7 per unit

5. Print the amount
6. Stop the program

Ex. No.:2

Currency Converter, Distance Converter and Time Converter

Aim:

To develop a Java application to implement currency converter , distance converter and time converter using packages.

Procedure:

1. Start the program
2. Create three packages for currency converter , distance converter and time converter.
3. Create corresponding code for conversion
4. Print the converted value.
5. Stop the program

Ex. No.:3

PAYROLL PROCESSING

Aim:

To develop a Java application with employee class and generate pay slips for the employees with their gross and net salary.

Procedure:

1. Start the program
2. Create Employee class with Emp_name, Emp_id, Address, Mail_id, Mobile_no as members.
3. Inherit the classes, Programmer, Assistant Professor, Associate Professor and Professor from employee class.
4. Add Basic Pay (BP) as the member of all the inherited classes with 97% of BP as DA, 10 % of BP as HRA, 12% of BP as PF, 0.1% of BP for staff club fund.
5. Generate pay slips for the employees with their gross and net salary.
6. Stop the program

Ex. No.:4

ADT STACK

Aim

To design a Java interface for ADT Stack using array.

Procedure:

1. Start the program
2. Define the interface.
3. Read the elements using array.
4. Initialize stackTop pointer as zero,
5. Define and use the method Push() to insert the elements into the stack with 'STACK OVERFLOW' condition.
6. Define and use the method pop() to remove an element from an array with 'STACK UNDERFLOW' condition
7. Display the output.

Ex. No.:5

STRING OPERATIONS

Aim

To write a program to perform string operations using ArrayList.

Procedure:

1. Start the program
2. Add the String as an object to List.
3. Get the choice from the user and do according to the choice
    a. Append-add at end
    b. Insert-add at particular index
    c. Search
    d. List all string starts with given letter.
3. Display the result
4. Stop the program.

Ex. No.:6

ABSTRACT CLASS

Aim

To write a Java Program to create an abstract class named Shape and provide three classes named Rectangle, Triangle and Circle such that each one of the classes extends the class Shape.

Procedure:

1. Start the program
2. Define the abstract class **shape**.
3. Define the class Rectangle with PrintArea() method that extends(makes use of) Shape.
4. Define the class Triangle with PrintArea() method that extends(makes use of) Shape.
5. Define the class Circle with PrintArea() method that extends(makes use of) Shape.
6. Print the area of the Rectangle,Triangle and Circle .
7. Stop the Program.

Ex. No.:7

EXCEPTION HANDLING

Aim

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

Procedure:

1. Start the program
2. Define the exception for getting a number from the user.
3. If the number is positive print the number as such.
4. If the number is negative throw the exception to the user as '**Number must be positive**'.
5. Stop the Program.

Ex. No.: 8

FILE INFORMATION

Aim

To write a Java program that reads a file name from the user, displays information about whether the file exists, whether the file is readable, or writable, the type of file and the length of the file in bytes.

Procedure:

1. Start the program
2. Read the filename from the user.
3. Use getName() Method to display the filename.
4. Use getPath() Method to display the path of the file.
5. Use getParent() Method to display its parent's information.
6. Use exists() Method to display whether the file exist or not
7. Use isFile() and isDirectory() Methods to display whether the file is file or directory.
8. Use canRead() and canWrite) methods to display whether the file is readable or writable.
9. Use lastModified() Method to display the modified information.
10. Use length() method to display the size of the file.
11. Use isHiddden() Method to display whether the file is hidden or not.

Ex. No.: 9

MULTITHREADING

Aim

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

Procedure:

1. Start the program
2. Design the first thread that generates a random integer for every 1 second .
3. If the first thread value is even, design the second thread as the square of the number and then print it.
4. If the first thread value is odd, then third thread will print the value of cube of the number.
5. Stop the program.

Ex. No.: 10

GENERIC FUNCTION

Aim

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

Procedure:

1. Start the program
2. Define the array with the elements
3. Sets the first value in the array as the current maximum
4. Find the maximum value by comparing each elements of the array
5. Display the maximum value
6. Stop the program.

Ex. No.: 11

CALCULATOR

Aim

To design a calculator using event-driven programming paradigm of Java for Decimal manipulations and Scientific manipulations.

Procedure:

1. Start the program
2. Using the swing components design the buttons of the calculator
3. Use key events and key listener to listen the events of the calculator.
4. Do the necessary manipulations.
5. Stop the program.