**JOURNAL-3**

1. **Write a program which accepts starting character and ending character. Display one by one character from starting character till the ending character at the interval of one second using thread.**

CODING

import java.util.Scanner;

public class CharacterThread implements Runnable {

private char startChar;

private char endChar;

public CharacterThread(char startChar, char endChar) {

this.startChar = startChar;

this.endChar = endChar;

}

@Override

public void run() {

try {

for (char c = startChar; c <= endChar; c++) {

System.out.print(c + " ");

Thread.sleep(1000);

}

} catch (InterruptedException e) {

e.printStackTrace();

}

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter starting character: ");

char startChar = scanner.next().charAt(0);

System.out.print("Enter ending character: ");

char endChar = scanner.next().charAt(0);

CharacterThread characterThread = new CharacterThread(startChar, endChar);

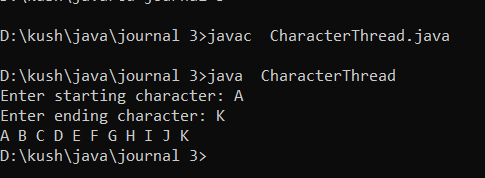
Thread thread = new Thread(characterThread);

thread.start();

}

}

OUTPUT



**2. Write a program that stores details of 5 employees and display this information after every 10 second.**

**CODING**

import java.util.ArrayList;

public class EmployeeDetails {

public static void main(String[] args) {

// Create an ArrayList to store employee objects

ArrayList<Employee> employees = new ArrayList<Employee>();

// Add 5 employees to the ArrayList

employees.add(new Employee("kush", "patel", "kp@gmail.com"));

employees.add(new Employee("rishi", "patel", "rp@gmail.com"));

employees.add(new Employee("jay", "panchal", "jp@gmail.com"));

employees.add(new Employee("keyur", "patel", "k@gmail.com"));

employees.add(new Employee("harsh", "panchal", "hp@gmail.com"));

// Create a new thread to display employee information every 10 seconds

Thread displayThread = new Thread(new Runnable() {

public void run() {

while (true) {

System.out.println("Employee Details:");

for (Employee emp : employees) {

System.out.println(emp);

}

System.out.println();

try {

Thread.sleep(10000); // Wait for 10 seconds

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

});

// Start the thread

displayThread.start();

}

}

// Employee class with fields for first name, last name, and email

class Employee {

String firstName;

String lastName;

String email;

public Employee(String firstName, String lastName, String email) {

this.firstName = firstName;

this.lastName = lastName;

this.email = email;

}

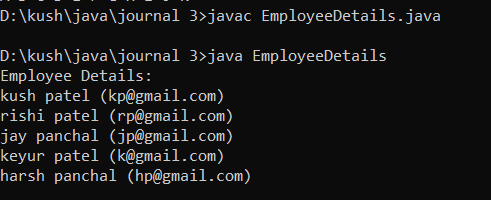
public String toString() {

return firstName + " " + lastName + " (" + email + ")";

}

}

**OUTPUT**

****

**3. Write a java application which accepts 10 names of student and their age. Sort names and age in descending order at an interval of 1 second using thread.**

**CODING**

import java.util.\*;

public class StudentSorter implements Runnable {

private String[] names;

private int[] ages;

public StudentSorter(String[] names, int[] ages) {

this.names = names;

this.ages = ages;

}

@Override

public void run() {

for (int i = 0; i < 10; i++) {

try {

Thread.sleep(1000); // wait for 1 second

} catch (InterruptedException e) {

e.printStackTrace();

}

sortDescending(); // sort the names and ages in descending order

System.out.println(Arrays.toString(names));

System.out.println(Arrays.toString(ages));

}

}

private void sortDescending() {

Map<String, Integer> map = new HashMap<>();

for (int i = 0; i < 10; i++) {

map.put(names[i], ages[i]); // create a map of names and ages

}

List<Map.Entry<String, Integer>> list = new ArrayList<>(map.entrySet());

Collections.sort(list, new Comparator<Map.Entry<String, Integer>>() {

public int compare(Map.Entry<String, Integer> o1, Map.Entry<String, Integer> o2) {

return (o2.getValue()).compareTo(o1.getValue()); // sort in descending order based on age

}

});

for (int i = 0; i < 10; i++) {

Map.Entry<String, Integer> entry = list.get(i);

names[i] = entry.getKey(); // update the sorted names array

ages[i] = entry.getValue(); // update the sorted ages array

}

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

String[] names = new String[10];

int[] ages = new int[10];

for (int i = 0; i < 10; i++) {

System.out.print("Enter name of student " + (i+1) + ": ");

names[i] = scanner.next();

System.out.print("Enter age of student " + (i+1) + ": ");

ages[i] = scanner.nextInt();

}

StudentSorter sorter = new StudentSorter(names, ages);

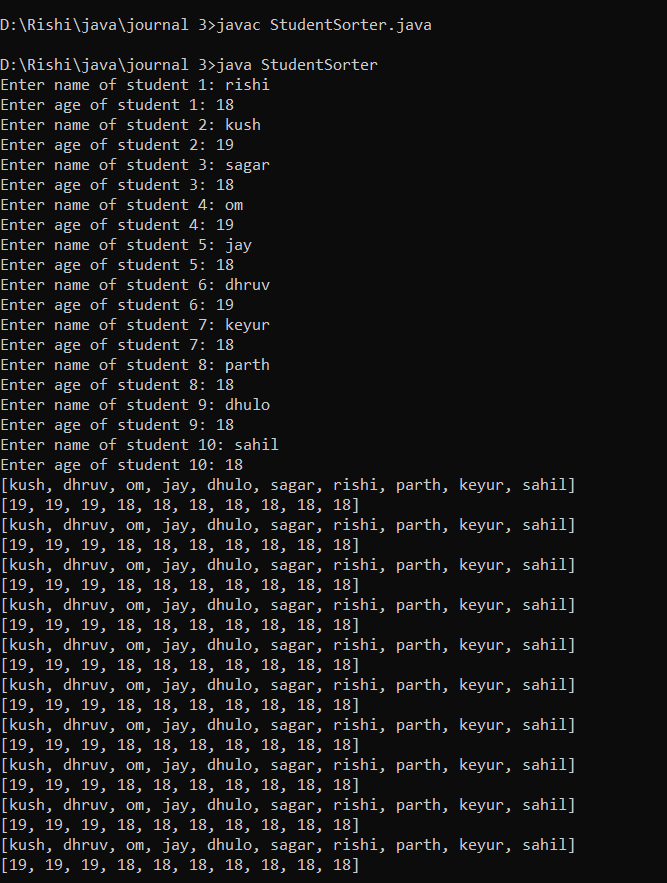
Thread thread = new Thread(sorter);

thread.start();

}

}

OUTPUT

****

**4. Create package stores. Under it create a class called stock with member variable (item\_no, item\_name, stock\_availible, and cost). Under the default package create a class called sales with field name (qty\_sold) and it is the child class of stores class. Write a program to print the current stock of each item and perform addition.**

**CODING**

package store;

public class stock {

private int item\_no;

private String item\_name;

private int stock\_available;

private double cost;

public stock(int item\_no, String item\_name, int stock\_available, double cost) {

this.item\_no = item\_no;

this.item\_name = item\_name;

this.stock\_available = stock\_available;

this.cost = cost;

}

public int getItem\_no() {

return item\_no;

}

public String getItem\_name() {

return item\_name;

}

public int getStock\_available() {

return stock\_available;

}

public double getCost() {

return cost;

}

public void setStock\_available(int stock\_available) {

this.stock\_available = stock\_available;

}

public void setCost(double cost) {

this.cost = cost;

}

public String toString() {

return "Item No.: " + item\_no + ", Item Name: " + item\_name + ", Stock Available: " + stock\_available + ", Cost: " + cost;

}

}

import store.\*;

import java.util.ArrayList;

import java.util.Scanner;

public class PRG\_04 {

public static void main(String[] args) {

ArrayList<stock> items = new ArrayList<stock>();

items.add(new stock(1, "Apple", 10, 20.0));

items.add(new stock(2, "Banana", 20, 30.0));

items.add(new stock(3, "Ball", 30, 40.0));

Scanner scanner = new Scanner(System.in);

while (true) {

System.out.println("\nCurrent Stock:");

for (stock item : items) {

System.out.println(item);

}

System.out.print("\nEnter the item no. to add stock, or 0 to exit:");

int item\_no = scanner.nextInt();

if (item\_no == 0) {

break;}

stock item = items.stream().filter(i -> i.getItem\_no()==item\_no).findFirst().orElse(null);

if (item == null) {

System.out.println("Invalid item no.");

}

else {

System.out.print("\nEnter the quantity to add:");

int qty = scanner.nextInt();

item.setStock\_available(item.getStock\_available() + qty);

System.out.println("Stock added successfully.");

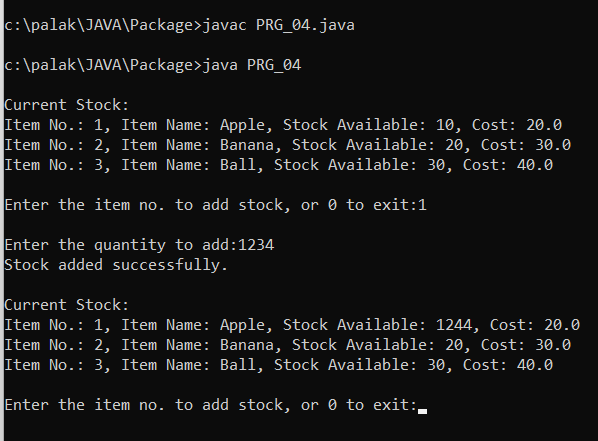
}

}

}

}

**OUTPUT**



**5. Create a class namely Vehicle to maintain vehicle data like chassisNo, nameOfVehicle, colour, owner using singly circular linked list. Perform following operations on student list: a. Add vehicle details at the end of the list b. Remove last vehicle detail in the list c. Display all vehicle details in the list**

**CODING**

import java.lang.\*;

class Vehicle {

private int chassisNo;

private String nameOfVehicle;

private String colour;

private String owner;

private Vehicle next;

public Vehicle(int chassisNo, String nameOfVehicle, String colour, String owner) {

this.chassisNo = chassisNo;

this.nameOfVehicle = nameOfVehicle;

this.colour = colour;

this.owner = owner;

this.next = null;

}

public void setNext(Vehicle next) {

this.next = next;

public Vehicle getNext() {

return next;

public void addVehicle(Vehicle vehicle) {

Vehicle current = this;

while (current.next == this) {

current = current.next;

}

current.next = vehicle;

vehicle.next = this;

}

public void removeLastVehicle() {

Vehicle current = this;

while (current.next.next != this) {

current = current.next;

}

current.next = this;

}

public void displayAllVehicles() {

Vehicle current = this;

do {

System.out.println("Chassis No: " + current.chassisNo);

System.out.println("Name of Vehicle: " + current.nameOfVehicle);

System.out.println("Colour: " + current.colour);

System.out.println("Owner: " + current.owner);

System.out.println("-------------------------");

current = current.next;

} while (current != this);

}

}

//MAIN METHOD:

public class vehicle\_detail {

public static void main(String[] args) {

Vehicle vehicle1 = new Vehicle(1234, "Car", "Red", "KUSH");

Vehicle vehicle2 = new Vehicle(5678, "Motorcycle", "Blue", "RISHI");

Vehicle vehicle3 = new Vehicle(9012, "Truck", "Green", "JAY");

vehicle1.addVehicle(vehicle2);

vehicle1.addVehicle(vehicle3);

vehicle1.displayAllVehicles()

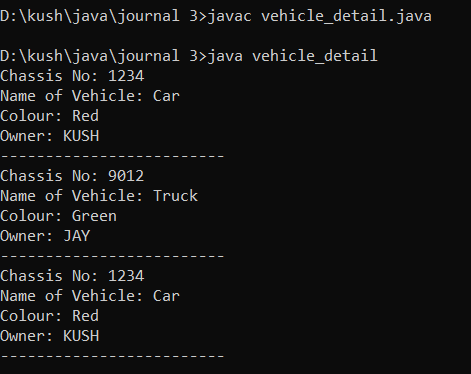
vehicle1.removeLastVehicle();

vehicle1.displayAllVehicles();

}

}

OUTPUT



**6. Create a class namely Book to maintain Book details like id, name, quantity and author using singly linked list. Perform following operations on book list: a. Add book details in the begging of the list b. Add book details at the end of the list c. Add book detail at particular position d. Remove first book detail from the list e. Remove last book detail from the list f. Display all book details in the list**

CODING

import java.util.Scanner;

class Book {

private int id;

private String name;

private int quantity;

private String author;

private Book next;

public Book(int id, String name, int quantity, String author) {

this.id = id;

this.name = name;

this.quantity = quantity;

this.author = author;

this.next = null;

System.out.println("\nData Inserted Successfully.");

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public int getQuantity() {

return quantity;

}

public void setQuantity(int quantity) {

this.quantity = quantity;

}

public String getAuthor() {

return author;

}

public void setAuthor(String author) {

this.author = author;

}

public Book getNext() {

return next;

}

public void setNext(Book next) {

this.next = next;

}

}

class BookList {

private Book head;

public BookList() {

head = null;

}

public void addBookAtBeginning(int id, String name, int quantity, String author) {

Book newBook = new Book(id, name, quantity, author);

newBook.setNext(head);

head = newBook;

}

public void addBookAtEnd(int id, String name, int quantity, String author) {

Book newBook = new Book(id, name, quantity, author);

if (head == null) {

head = newBook;

} else {

Book current = head;

while (current.getNext() != null) {

current = current.getNext();

}

current.setNext(newBook);

}

}

public void addBookAtPosition(int id, String name, int quantity, String author, int position) {

if (position == 1) {

addBookAtBeginning(id, name, quantity, author);

} else {

Book newBook = new Book(id, name, quantity, author);

Book current = head;

int currentPosition = 1;

while (currentPosition < position - 1 && current != null) {

current = current.getNext();

currentPosition++;

}

if (current != null) {

newBook.setNext(current.getNext());

current.setNext(newBook);

} else {

System.out.println("Invalid position");

}

}

}

public void removeFirstBook() {

if (head == null) {

System.out.println("List is empty");

} else {

head = head.getNext();

}

}

public void removeLastBook() {

if (head == null) {

System.out.println("List is empty");

}

else if (head.getNext() == null) {

head = null;

}

else {

Book current =head;

while (current.getNext().getNext() != null) {

current = current.getNext();

}

current.setNext(null);

}

}

public void displayBooks() {

if (head == null) {

System.out.println("List is empty");

} else {

Book current = head;

System.out.println("-----------------------------------------------");

while (current != null) {

System.out.println("ID: " + current.getId() + ", Name: " + current.getName() + ", Quantity: " + current.getQuantity() + ", Author: " + current.getAuthor());

current = current.getNext();

}

System.out.println("-----------------------------------------------");

}

}

}

public class BOOK\_DETAIL\_01

{

public static void main(String[] args)

{

int id;

String name;

int quantity;

String author;

Scanner scan = new Scanner(System.in);

BookList bookList = new BookList();

while (true) {

System.out.println("\n-----------------------------------------------");

System.out.println("\nSingly Linked List Operations\n");

System.out.println("-----------------------------------------------");

System.out.println("1. Insert at Begining.");

System.out.println("2. Insert at End.");

System.out.println("3. Insert at Position.");

System.out.println("4. Delete from Head.");

System.out.println("5. Delete from Tail.");

System.out.println("6. Display Data.");

System.out.println("7. Exit.");

System.out.println("-----------------------------------------------");

System.out.print("Enter your Choice : ");

int choice = scan.nextInt();

switch (choice)

{

case 1 :

System.out.print("Enter Your ID : ");

id=scan.nextInt();

scan.nextLine();

System.out.print("Enter Your Name : ");

name=scan.nextLine();

System.out.print("Enter Quantity of Books : ");

quantity=scan.nextInt();

scan.nextLine();

System.out.print("Enter Author Name : ");

author=scan.nextLine();

bookList.addBookAtBeginning(id,name,quantity,author);

break;

case 2 :

System.out.print("Enter Your ID : ");

id=scan.nextInt();

scan.nextLine();

System.out.print("Enter Your Name : ");

name=scan.nextLine();

System.out.print("Enter Quantity of Books : ");

quantity=scan.nextInt();

scan.nextLine();

System.out.print("Enter Author Name : ");

author=scan.nextLine();

bookList.addBookAtEnd(id,name,quantity,author);

break;

case 3 :

int position;

System.out.print("Enter Position you want to Insert Record : ");

position=scan.nextInt();

System.out.print("Enter Your ID : ");

id=scan.nextInt();

scan.nextLine();

System.out.print("Enter Your Name : ");

name=scan.nextLine();

System.out.print("Enter Quantity of Books : ");

quantity=scan.nextInt();

scan.nextLine();

System.out.print("Enter Author Name : ");

author=scan.nextLine();

bookList.addBookAtPosition(id,name,quantity,author,position);

break;

case 4 :

bookList.removeFirstBook();

System.out.println("\nData Deleted Successfully.");

break;

case 5 :

bookList.removeLastBook();

System.out.println("\nData Deleted Successfully.");

break;

case 6 :

bookList.displayBooks();

break;

case 7 :

System.out.println("Program Exited...");

System.exit(0);

break;

default :

System.out.println("Invalid choice. Try again.");

break;

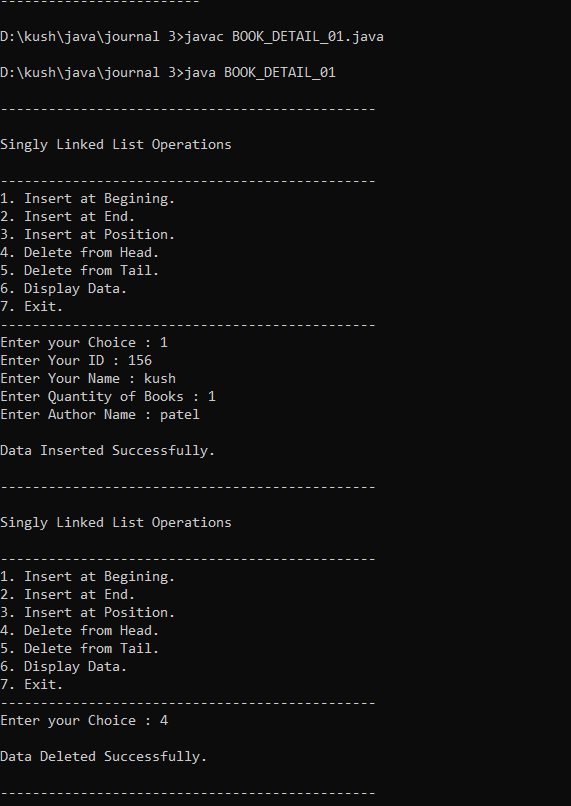
}

}

}

}

OUTPUT



**7. Write a programme to draw smiley with colour using applet.**

**CODING**

import java.applet.Applet;

import java.awt.Color;

import java.awt.Graphics;

public class Smiley extends Applet {

/\*<applet code="Smiley.class" width="100" height="100"></applet>\*/

public void paint(Graphics g) {

// draw face

g.setColor(Color.YELLOW);

g.fillOval(50, 50, 200, 200);

g.setColor(Color.BLACK);

g.drawOval(50, 50, 200, 200);

// draw eyes

g.setColor(Color.BLUE);

g.fillOval(100, 100, 25, 25);

g.fillOval(175, 100, 25, 25);

// draw mouth

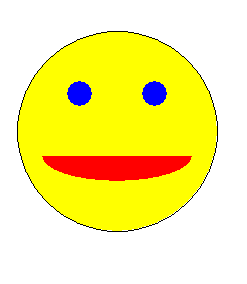
g.setColor(Color.RED);

g.fillArc(75, 150, 150, 50, 180, 180);

}

}

**OUTPUT**



**8. Create an applet which displays a solid square having red colour. Display name of your college within the square with font style =’Times New Roman’, font size=12 and font colour=’Yellow’.**

CODING

import java.applet.Applet;

import java.awt.Color;

import java.awt.Font;

import java.awt.Graphics;

public class CollegeSquare extends Applet {

/\*<applet code="CollegeSquare.class" width="100" height="100"></applet>\*/

public void paint(Graphics g) {

// draw square

g.setColor(Color.cyan);

g.fillRect(50, 50, 200, 200);

// set font

Font font = new Font("Times New Roman", Font.PLAIN, 12);

g.setFont(font);

// draw text

g.setColor(Color.red);

g.drawString("AGRAWAL PALAK", 70, 130);

}

}

**OUTPUT**



**9. Write a program to draw circle inside a square in applet with different colours**

CODING

import java.applet.Applet;

import java.awt.Color;

import java.awt.Graphics;

public class CircleSquare extends Applet {

/\*<applet code="CircleSquare.class" width="100" height="100"></applet>\*/

public void paint(Graphics g) {

// draw square

g.setColor(Color.BLUE);

g.fillRect(50, 50, 200, 200);

// draw circle

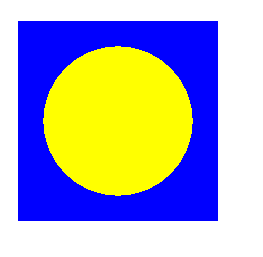
g.setColor(Color.YELLOW);

g.fillOval(75, 75, 150, 150);

}

}

**OUTPUT**

****

**10. Write an applet program which accepts number of ovals user wants to display using parameter tag and draws ovals in different positions.**

CODING

import java.applet.Applet;

import java.awt.Color;

import java.awt.Graphics;

import java.util.Random;

public class OvalApplet extends Applet {

/\* <applet code="OvalApplet.class" width="500" height="500">

<param name="numOvals" value="5">

</applet>\*/

int numOvals;

public void init() {

String num = getParameter("numOvals");

numOvals = Integer.parseInt(num);

}

public void paint(Graphics g) {

for(int i = 1; i <= numOvals; i++) {

int x = (int)(Math.random() \* getWidth());

int y = (int)(Math.random() \* getHeight());

int width = (int)(Math.random() \* 100);

int height = (int)(Math.random() \* 100);

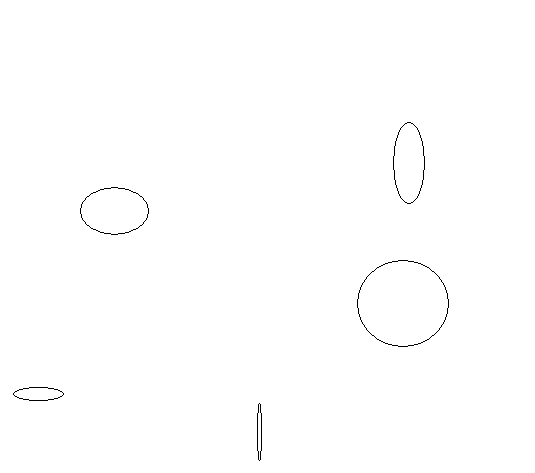
g.drawOval(x, y, width, height);

}

}

}

**OUTPUT**

****