

# JOURNAL-03

21BCA135



PANCHAL JAY K.

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.

```
import java.util.Scanner;
```

```
public class PRG_01 {
```

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

```
        Scanner scanner = new Scanner(System.in);
```

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

```
        char startChar = scanner.nextLine().charAt(0);
```

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

```
        char endChar = scanner.nextLine().charAt(0);
```

```
        CharacterDisplayThread thread = new CharacterDisplayThread(startChar, endChar);
```

```
        thread.start();
```

```
    }
```

```
}
```

```
class CharacterDisplayThread extends Thread {
```

```
    private char startChar;
```

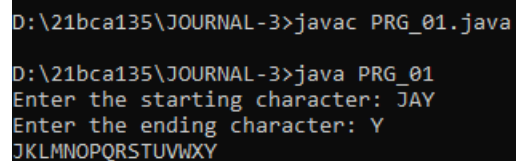
```
    private char endChar;
```

```

public CharacterDisplayThread(char startChar, char endChar) {
    this.startChar = startChar;
    this.endChar = endChar;
}
public void run() {
    for (char ch = startChar; ch <= endChar; ch++) {
        System.out.print(ch);
        try {
            Thread.sleep(1000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
}

```

Output :



```

D:\21bca135\JOURNAL-3>javac PRG_01.java
D:\21bca135\JOURNAL-3>java PRG_01
Enter the starting character: JAY
Enter the ending character: Y
JKLMNOPQRSTUVWXYZ

```

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

```

import java.util.Scanner;

public class PRG_02 {

    public static void main(String[] args) {

```

```

String[] name=new String[5];
int[] age=new int[5];
String[] department=new String[5];;
double[] salary=new double[5];
Scanner sc = new Scanner(System.in);
for(int i=0;i<5;i++)
{

    System.out.print("Enter Emp "+ (i+1) +" Name : ");
    name[i] = sc.nextLine();
    System.out.print("Enter Emp "+ (i+1) +" Age : ");
    age[i] = sc.nextInt();
    sc.nextLine();
    System.out.print("Enter Emp "+ (i+1) +" Department : ");
    department[i] = sc.nextLine();
    System.out.print("Enter Emp "+ (i+1) +" Salary : ");
    salary[i] = sc.nextDouble();
    sc.nextLine();
    System.out.println();
}

for(int i=0;i<5;i++)
{

    try {

        System.out.print("\nName: " + name[i] + ", Age: " + age[i] + ",
        Department: " + department[i] + ", Salary: " + salary[i]);
    }
}

```

```

        Thread.sleep(10000);
    }
    catch (InterruptedException e)
    {
        e.printStackTrace();
    }
}
}
}

```

### Output :

```

D:\21bca135\JOURNAL-3>javac PRG_02.java
D:\21bca135\JOURNAL-3>java PRG_02
Enter Emp 1 Name : jay
Enter Emp 1 Age : 20
Enter Emp 1 Department : bca
Enter Emp 1 Salary : 4500

Enter Emp 2 Name : krish
Enter Emp 2 Age : 1000
Enter Emp 2 Department : bcom
Enter Emp 2 Salary : 45.

Enter Emp 3 Name : kur
Enter Emp 3 Age : 10
Enter Emp 3 Department : mca
Enter Emp 3 Salary : 5

Enter Emp 4 Name : harsh
Enter Emp 4 Age : 45
Enter Emp 4 Department : bba
Enter Emp 4 Salary : 4550

Enter Emp 5 Name : jay
Enter Emp 5 Age : 20
Enter Emp 5 Department : mca
Enter Emp 5 Salary : 50000

Name: jay, Age: 20, Department: bca, Salary: 4500.0
Name: krish, Age: 1000, Department: bcom, Salary: 45.0
Name: kur, Age: 10, Department: mca, Salary: 5.0
Name: harsh, Age: 45, Department: bba, Salary: 4550.0
Name: jay, Age: 20, Department: mca, Salary: 50000.0

```

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.

```
import java.util.Arrays;

import java.util.Scanner;

public class PRG_03 {

    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.nextLine();

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

            ages[i] = scanner.nextInt();

            scanner.nextLine();

        }

        while (true) {

            System.out.println("\nSelect an option:");
```

```

System.out.println("1. Sort via Name.");
System.out.println("2. Sort via Age.");
System.out.println("3. Exit");
System.out.print("\nSelect Your Choice : ");
int choice = scanner.nextInt();
scanner.nextLine();
switch (choice) {
    case 1:
        for (int i = 0; i < 10; i++) {
            for (int j = i + 1; j < 10; j++) {
                if (names[i].compareToIgnoreCase(names[j]) < 0) {
                    String tempName = names[i];
                    names[i] = names[j];
                    names[j] = tempName;
                    int tempAge = ages[i];
                    ages[i] = ages[j];
                    ages[j] = tempAge;
                }
            }
        }
        System.out.println("\nSorted Names in Descending Order:");
        for (int i = 0; i < 10; i++) {

```

```

        try {
            System.out.println(names[i] + " - " + ages[i]);
            Thread.sleep(1000);
        }
        catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
    break;
case 2:
    for (int i = 0; i < 10; i++) {
        for (int j = i + 1; j < 10; j++) {
            if (ages[i] < (ages[j])) {
                int tempage = ages[i];
                ages[i] = ages[j];
                ages[j] = tempage;
                String tempname = names[i];
                names[i] = names[j];
                names[j] = tempname;
            }
        }
    }
}

```



```

        System.out.println("\nSorted Ages in Descending Order:");
        for (int i = 0; i < 10; i++) {
            try {
                System.out.println(ages[i] + " - " + names[i]);
                Thread.sleep(1000);
            }
            catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
        break;
    case 3:
        System.out.println("Exiting program...");
        System.exit(0);
        break;
    default:
        System.out.println("Invalid choice. Try again.");
    }
}
}
}
}

```

## Output :

```
D:\21bca135\JOURNAL-3>javac PRG_03.java

D:\21bca135\JOURNAL-3>java PRG_03
Enter name of student 1: jay
Enter age of student 1: 10
Enter name of student 2: krrish
Enter age of student 2: 5
Enter name of student 3: ashish
Enter age of student 3: 100
Enter name of student 4: yagni
Enter age of student 4: 5
Enter name of student 5: kur
Enter age of student 5:
44
Enter name of student 6: harsh
Enter age of student 6: 10
Enter name of student 7: jay
Enter age of student 7: 23
Enter name of student 8: krupa
Enter age of student 8: 20
Enter name of student 9: shruti
Enter age of student 9: 10
Enter name of student 10: isha
Enter age of student 10: 20

Select an option:
1. Sort via Name.
2. Sort via Age.
3. Exit
```

```
Select Your Choice : 1

Sorted Names in Descending Order:
yagni - 5
shruti - 10
kur - 44
krupa - 20
krrish - 5
jay - 10
jay - 23
isha - 20
harsh - 10
ashish - 100
```

```
Select an option:
1. Sort via Name.
2. Sort via Age.
3. Exit
```

```
Select Your Choice : 2

Sorted Ages in Descending Order:
100 - ashish
44 - kur
23 - jay
20 - isha
20 - krupa
10 - jay
10 - harsh
10 - shruti
5 - krrish
5 - yagni
```

```
Select an option:
1. Sort via Name.
2. Sort via Age.
3. Exit
```

```
Select Your Choice : 3
Exiting program...
```

4. Create package stores. Under it create a class called stock with member variable (item\_no, item\_name, stock\_available, 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.

```
import stores.stock;
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 :

```
D:\21bca135\JOURNAL-3>javac PRG_04.java
D:\21bca135\JOURNAL-3>java PRG_04

Current Stock:
Item No.: 1, Item Name: Apple, Stock Available: 10, Cost: 20.0
Item No.: 2, Item Name: Banana, Stock Available: 20, Cost: 30.0
Item No.: 3, Item Name: Ball, Stock Available: 30, Cost: 40.0

Enter the item no. to add stock, or 0 to exit:3

Enter the quantity to add:45
Stock added successfully.

Current Stock:
Item No.: 1, Item Name: Apple, Stock Available: 10, Cost: 20.0
Item No.: 2, Item Name: Banana, Stock Available: 20, Cost: 30.0
Item No.: 3, Item Name: Ball, Stock Available: 75, Cost: 40.0
```

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.

```
import java.util.Scanner;

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

public int getChassisNo() {
    return chassisNo;
}

public void setChassisNo(int chassisNo) {
    this.chassisNo = chassisNo;
}

public String getNameOfVehicle() {
    return nameOfVehicle;
}

public void setNameOfVehicle(String nameOfVehicle) {
    this.nameOfVehicle = nameOfVehicle;
}
```

```
public String getColour() {  
    return colour;  
}
```

```
public void setColour(String colour) {  
    this.colour = colour;  
}
```

```
public String getOwner() {  
    return owner;  
}
```

```
public void setOwner(String owner) {  
    this.owner = owner;  
}
```

```
public Vehicle getNext() {  
    return next;  
}
```

```
public void setNext(Vehicle next) {  
    this.next = next;  
}  
}
```

```
class VehicleList {  
    private Vehicle tail;
```



```

public VehicleList() {
    tail = null;
}

public void addVehicle(int chassisNo, String nameOfVehicle, String colour,
String owner) {
    Vehicle newVehicle = new Vehicle(chassisNo, nameOfVehicle, colour,
owner);
    if (tail == null) {
        tail = newVehicle;
        tail.setNext(tail);
    }
    else {
        newVehicle.setNext(tail.getNext());
        tail.setNext(newVehicle);
        tail = newVehicle;
    }
}

public void removeLastVehicle() {
    if (tail == null) {
        System.out.println("List is empty");
        return;
    }
    if (tail.getNext() == tail) {
        tail = null;
        return;
    }
}

```

```

        Vehicle current = tail.getNext();
        while (current.getNext() != tail) {
            current = current.getNext();
        }
        current.setNext(tail.getNext());
        tail = current;
    }
    public void displayVehicles() {
        if (tail == null) {
            System.out.println("List is empty");
            return;
        }
        Vehicle current = tail.getNext();
        do {
            System.out.println("-----");
            System.out.println("Chassis No: " + current.getChassisNo() +
                "\nName of Vehicle: " + current.getNameOfVehicle() +
                "\nColour: " + current.getColour() +
                "\nOwner: " + current.getOwner());
            System.out.println("-----");
            current = current.getNext();
        } while (current != tail.getNext());
    }
}

```

```

public class PRG_05 {
    public static void main(String[] args)

```

```

{
Scanner scan = new Scanner(System.in);
VehicleList vehicleList = new VehicleList();
while (true) {
    System.out.println("\n-----");
    System.out.println("\nCircular Singly Linked List Operations\n");
    System.out.println("-----");
    System.out.println("1. Insert at End.");
    System.out.println("2. Delete from End.");
    System.out.println("3. Get Item detail's.");
    System.out.println("4. Exit.");
    System.out.println("-----");
    System.out.print("Enter your Choice : ");
    int choice = scan.nextInt();
    switch (choice)
    {
        case 1 :
            int ch_no;
            String nameOfVeh, colour, owner;
            System.out.print("Enter Chassis_No : ");
            ch_no=scan.nextInt();
            scan.nextLine();
            System.out.print("Enter Name of vehicle : ");
            nameOfVeh=scan.nextLine();
            System.out.print("Enter Color of vehicle : ");
            colour=scan.nextLine();
            System.out.print("Enter Owner Name : ");
            owner=scan.nextLine();

```

```

        vehicleList.addVehicle(ch_no,nameOfVeh,colour,owner);
        break;
case 2 :
    vehicleList.removeLastVehicle();
    System.out.println("\nData Deleted Successfully.");
    break;
case 3 :
    System.out.println("Vehicle details:");
    vehicleList.displayVehicles();
    break;
case 4 :
    System.out.println("Program Exited...");
    System.exit(0);
    break;
default:
    System.out.println("Invalid choice. Try again.");
    }
    }
    }
}

```

**Output :**

```
D:\21bca135\JOURNAL-3>javac PRG_05.java
```

```
D:\21bca135\JOURNAL-3>java PRG_05
```

```
-----  
Circular Singly Linked List Operations  
-----
```

1. Insert at End.
  2. Delete from End.
  3. Get Item detail's.
  4. Exit.
- ```
-----
```

```
Enter your Choice : 1  
Enter Chassis_No : 123  
Enter Name of vehicle : royal enfield  
Enter Color of vehicle : red  
Enter Owner Name : jay
```

```
Data Inserted Successfully.  
-----
```

```
Circular Singly Linked List Operations  
-----
```

1. Insert at End.
  2. Delete from End.
  3. Get Item detail's.
  4. Exit.
- ```
-----
```

```
Enter your Choice : 2
```

```
Data Deleted Successfully.  
-----
```

### Circular Singly Linked List Operations

1. Insert at End.
2. Delete from End.
3. Get Item detail's.
4. Exit.

Enter your Choice : 3

Vehicle details:

Chassis No: 1235454

Name of Vehicle: royal enfield

Colour: black

Owner: jay panchal

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 beginning 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 .

```
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 PRG_06  
{  
    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 Beginning.");  
            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 :**

```
D:\21bca135\JOURNAL-3>javac PRG_06.java
```

```
D:\21bca135\JOURNAL-3>java PRG_06
```

```
-----  
Singly Linked List Operations  
-----
```

1. Insert at Begining.
2. Insert at End.
3. Insert at Position.
4. Delete from Head.
5. Delete from Tail.
6. Display Data.
7. Exit.

```
-----  
Enter your Choice : 1
```

```
Enter Your ID : 135
```

```
Enter Your Name : jay panchal
```

```
Enter Quantity of Books : 4
```

```
Enter Author Name : jay
```

```
Data Inserted Successfully.  
-----
```

```
Singly Linked List Operations  
-----
```

1. Insert at Begining.
2. Insert at End.
3. Insert at Position.
4. Delete from Head.
5. Delete from Tail.
6. Display Data.
7. Exit.

```
-----  
Enter your Choice : 2
```

```
Enter Your ID : 134
```

```
Enter Your Name : harsh
```

```
Enter Quantity of Books : 3
```

```
Enter Author Name : harsh
```

```
Data Inserted Successfully.
```



### Singly Linked List Operations

- 1. Insert at Beginning.
- 2. Insert at End.
- 3. Insert at Position.
- 4. Delete from Head.
- 5. Delete from Tail.
- 6. Display Data.
- 7. Exit.

Enter your Choice : 3

Enter Position you want to Insert Record : 2

Enter Your ID : 11

Enter Your Name : kur

Enter Quantity of Books : 10

Enter Author Name : kur

Data Inserted Successfully.

### Singly Linked List Operations

- 1. Insert at Beginning.
- 2. Insert at End.
- 3. Insert at Position.
- 4. Delete from Head.
- 5. Delete from Tail.
- 6. Display Data.
- 7. Exit.

Enter your Choice : 4

Data Deleted Successfully.

### Singly Linked List Operations

- 1. Insert at Beginning.
- 2. Insert at End.
- 3. Insert at Position.
- 4. Delete from Head.
- 5. Delete from Tail.

```
Enter your Choice : 5
```

```
Data Deleted Successfully.
```

```
-----  
Singly Linked List Operations
```

- ```
-----  
1. Insert at Begining.  
2. Insert at End.  
3. Insert at Position.  
4. Delete from Head.  
5. Delete from Tail.  
6. Display Data.  
7. Exit.  
-----
```

```
Enter your Choice : 6
```

```
-----  
ID: 11, Name: kur, Quantity: 10, Author: kur  
-----  
-----
```

```
Singly Linked List Operations
```

- ```
-----  
1. Insert at Begining.  
2. Insert at End.  
3. Insert at Position.  
4. Delete from Head.  
5. Delete from Tail.  
6. Display Data.  
7. Exit.  
-----
```

```
Enter your Choice : 7
```

```
Program Exited...
```

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

```
import java.awt.*;
import java.applet.*;

//<applet code="PRG_07.class" height="800" width="1860"> </applet>

public class PRG_07 extends Applet {

    public void paint(Graphics g) {

        g.setColor(Color.yellow);
        g.fillOval(50,50,200,200);
        g.setColor(Color.black);
        g.drawOval(50,50,200,200);

        g.setColor(Color.black);
        g.fillOval(100,100,25,25);
        g.fillOval(175,100,25,25);

        g.setColor(Color.black);
        g.drawArc(100,125,100,75,0,-180);

    }
}
```

Output :

```
D:\21bca135\JOURNAL-3>javac PRG_07.java  
D:\21bca135\JOURNAL-3>appletviewer PRG_07.JAVA
```



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=50 and font colour='Yellow'.

```
import java.awt.*;
import java.applet.*;

//<applet code="PRG_08.class" height="800" width="1860"> </applet>

public class PRG_08 extends Applet {

    public void paint(Graphics g) {

        g.setColor(Color.red);
        g.fillRect(200,200,400,400);

        g.setColor(Color.yellow);
        Font font = new Font("Times New Roman", Font.PLAIN, 50);
        g.setFont(font);
        FontMetrics metrics = g.getFontMetrics(font);
        int x = (200 - metrics.stringWidth("My College")) / 2;
        int y = ((200 - metrics.getHeight()) / 2) + metrics.getAscent();
        g.drawString("VTCBCSR", 300+x, 300+y);

    }
}
```

## Output :

```
D:\21bca135\JOURNAL-3>javac PRG_08.java
D:\21bca135\JOURNAL-3>appletviewer PRG_08.JAVA
```



VTCBCSR

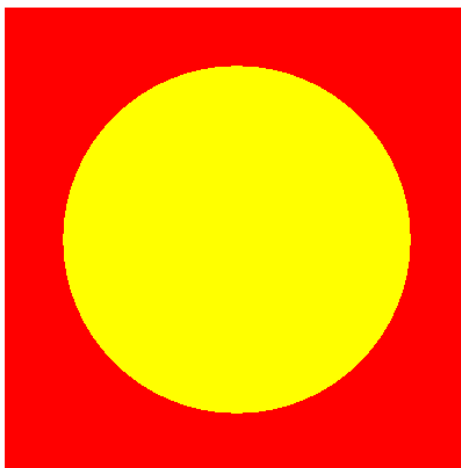
9. Write a program to draw circle inside a square in applet with different colours.

```
import java.awt.*;  
import java.applet.*;  
  
//<applet code="PRG_09.class" height="800" width="1860"> </applet>  
  
public class PRG_09 extends Applet {  
  
    public void paint(Graphics g) {
```

```
g.setColor(Color.red);  
g.fillRect(200,200,400,400);  
  
g.setColor(Color.yellow);  
g.fillOval(250,250,300,300);  
  
}  
  
}
```

Output :

```
D:\21bca135\JOURNAL-3>javac PRG_09.java  
D:\21bca135\JOURNAL-3>appletviewer PRG_09.JAVA
```



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

```
import java.awt.*;
import java.applet.*;

/*<applet code="PRG_10.class" height="800" width="1860">
<param name="numOvals" value="10">
</applet>*/

public class PRG_10 extends Applet {
    private int numOvals;

    public void init() {
        String numOvalsStr = getParameter("numOvals");
        numOvals = Integer.parseInt(numOvalsStr);
    }

    public void paint(Graphics g) {
        for (int i = 0; i < numOvals; i++) {
            int x = (int)(Math.random() * 300);
            int y = (int)(Math.random() * 300);
            int w = (int)(Math.random() * 100);
            int h = (int)(Math.random() * 100);
            g.drawOval(x, y, w, h);
        }
    }
}
```



## Output :

```
D:\21bca135\JOURNAL-3>javac PRG_10.java  
D:\21bca135\JOURNAL-3>appletviewer PRG_10.JAVA
```

