PRACTICAL 23

AIM: Write a java program for producer and consumer problems using Threads.

```
class Buffer {
   private int data;
   private boolean available = false;

   public synchronized void put(int value) {
       while (available) {
           try {
               wait();
           } catch (InterruptedException e) {
               e.printStackTrace();
           }
       }
       data = value;
       available = true;
       notify();
   }

   public synchronized int get() {
       while (!available) {
           try {
               wait();
           } catch (InterruptedException e) {
               e.printStackTrace();
           }
       }
       available = false;
       notify();
       return data;
   }
}

class Producer extends Thread {
   private Buffer buffer;
```

```
    public Producer(Buffer buffer) {
        this.buffer = buffer;
    }

    @Override
    public void run() {
        for (int i = 0; i < 5; i++) {
            buffer.put(i);
            System.out.println("Produced: " + i);
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}

class Consumer extends Thread {
    private Buffer buffer;

    public Consumer(Buffer buffer) {
        this.buffer = buffer;
    }

    @Override
    public void run() {
        for (int i = 0; i < 5; i++) {
            int data = buffer.get();
            System.out.println("Consumed: " + data);
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}

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

```
        Buffer buffer = new Buffer();
        Producer producer = new Producer(buffer);
        Consumer consumer = new Consumer(buffer);

        producer.start();
        consumer.start();
    }
}
```

Output:

Produced: 0
Consumed: 0
Consumed: 1
Produced: 1
Produced: 2
Consumed: 2
Consumed: 3
Produced: 3
Consumed: 4
Produced: 4

Practical No: 24

AIM: Write a java program that implements a multi-thread application that has three threads. First thread generates a random integer every 1 second and if the value is even, the second thread computes the square of the number and prints. If the value is odd, the third thread will print the value of the cube of the number.

```java
import java.util.Random;

class GeneratorThread extends Thread {
    private Random random;
    private EvenThread evenThread;
    private OddThread oddThread;

    public GeneratorThread(EvenThread evenThread, OddThread
oddThread) {
        this.random = new Random();
        this.evenThread = evenThread;
        this.oddThread = oddThread;
    }

    @Override
    public void run() {
        while (true) {
            int number = random.nextInt(100);
            System.out.println("Generated number: " + number);
            if (number % 2 == 0) {
                evenThread.setNumber(number);
            } else {
                oddThread.setNumber(number);
            }
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}

class EvenThread extends Thread {
    private int number;

    @Override
    public void run() {
        while (true) {
            synchronized (this) {
                try {
```

```
                    wait();
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
                if (number != 0) {
                    System.out.println("Square of " + number + "
is: " + number * number);
                    number = 0;
                }
                this.notify(); // notify should be called on the
current object
            }
        }
    }

    public void setNumber(int number) {
        synchronized (this) { // synchronize on this object
            this.number = number;
            this.notify(); // notify should be called on the
current object
        }
    }
}

class OddThread extends Thread {
    private int number;

    @Override
    public void run() {
        while (true) {
            synchronized (this) {
                try {
                    wait();
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
                if (number != 0) {
                    System.out.println("Cube of " + number + "
is: " + number * number * number);
                    number = 0;
                }
```

```
                this.notify(); // notify should be called on the
current object
            }
        }
    }

    public void setNumber(int number) {
        synchronized (this) { // synchronize on this object
            this.number = number;
            this.notify(); // notify should be called on the
current object
        }
    }
}

public class MultiThreadedApplication {
    public static void main(String[] args) {
        EvenThread evenThread = new EvenThread();
        OddThread oddThread = new OddThread();
        GeneratorThread generatorThread = new
GeneratorThread(evenThread, oddThread);

        evenThread.start();
        oddThread.start();
        generatorThread.start();
    }
}
```

Output:

Generated number: 4
Square of 4 is: 16
Generated number: 98
Square of 98 is: 9604
Generated number: 31
Cube of 31 is: 29791
Generated number: 87
Cube of 87 is: 658503

…….. Etc.

Practical No: 25

AIM: write a program to create a dynamic array using the ArrayList class and then print the contents of the array object.

```java
import java.util.ArrayList;

public class DynamicArrayExample {
    public static void main(String[] args) {
        // Create an ArrayList (Dynamic Array)
        ArrayList<Integer> dynamicArray = new ArrayList<>();

        // Add elements to the ArrayList
        dynamicArray.add(10);
        dynamicArray.add(20);
        dynamicArray.add(30);
        dynamicArray.add(40);
        dynamicArray.add(50);

        // Print the contents of the ArrayList
        System.out.println("Contents of the dynamic array:");
        for (int element : dynamicArray) {
            System.out.println(element);
        }
    }
}
```

Output:

Contents of the dynamic array:

```
10
20
30
40
50

Process finished with exit code 0
```

Practical No: 26

AIM: Write programs to implement add, search and remove operations on ArrayList objects.

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

public class ArrayListOperations {
   private ArrayList<Integer> arrayList = new ArrayList<>();

   public void addElement(int element) {
       arrayList.add(element);
       System.out.println("Element " + element + " added to the
ArrayList.");
   }

   public boolean searchElement(int element) {
       return arrayList.contains(element);
```

```
    }

    public void removeElement(int element) {
        if (arrayList.contains(element)) {
            arrayList.remove((Integer) element);
            System.out.println("Element " + element + " removed
from the ArrayList.");
        } else {
            System.out.println("Element " + element + " not found
in the ArrayList.");
        }
    }

    public void displayArrayList() {
        System.out.println("ArrayList: " + arrayList);
    }

    public static void main(String[] args) {
        ArrayListOperations arrayListOperations = new
ArrayListOperations();
        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("1. Add element");
            System.out.println("2. Search element");
            System.out.println("3. Remove element");
            System.out.println("4. Display ArrayList");
            System.out.println("5. Exit");
            System.out.print("Enter your choice: ");
            int choice = scanner.nextInt();

            switch (choice) {
                case 1:
                    System.out.print("Enter element to add: ");
                    int addElement = scanner.nextInt();
                    arrayListOperations.addElement(addElement);
                    break;
                case 2:
                    System.out.print("Enter element to search:
");
                    int searchElement = scanner.nextInt();
```

```
                    boolean found =
arrayListOperations.searchElement(searchElement);
                    if (found) {
                        System.out.println("Element " +
searchElement + " found in the ArrayList.");
                    } else {
                        System.out.println("Element " +
searchElement + " not found in the ArrayList.");
                    }
                    break;
                case 3:
                    System.out.print("Enter element to remove:
");
                    int removeElement = scanner.nextInt();

arrayListOperations.removeElement(removeElement);
                    break;
                case 4:
                    arrayListOperations.displayArrayList();
                    break;
                case 5:
                    System.exit(0);
                    break;
                default:
                    System.out.println("Invalid choice. Please
try again.");
            }
        }
    }
}
```

1. Add element
2. Search element
3. Remove element
4. Display ArrayList
5. Exit

Enter your choice: 1
Enter element to add: 50
Element 50 added to the ArrayList.

1. Add element
2. Search element
3. Remove element
4. Display ArrayList
5. Exit

Enter your choice: 1
Enter element to add: 70
Element 70 added to the ArrayList.

1. Add element
2. Search element
3. Remove element
4. Display ArrayList
5. Exit

1. Add element
2. Search element
3. Remove element
4. Display ArrayList
5. Exit

Enter your choice: 3
Enter element to remove: 70
Element 70 removed from the ArrayList.

1. Add element
2. Search element
3. Remove element
4. Display ArrayList
5. Exit

Enter your choice: 4
ArrayList: [50]
1. Add element

2. Search element
3. Remove element
4. Display ArrayList
5. Exit
Enter your choice: 5

Process finished with exit code 0