WEEK 8: Implement program(s) to thread synchronization. (lab date : 06.10.25 - 10 programs)

1.Java program for extending Thread class

```
Java Thread Example by extending Thread class  class Multi extends Thread
{
public void run()
{ System.out.println("thread is running..."); }  public static void main(String args[])
{
Multi t1=new Multi();  t1.start();
}
}
```

Output:
thread is running...

2. Java program implementing Runnable interface

```
class Multi3 implements Runnable{
public void run(){  System.out.println("thread is running...");
}
public static void main(String args[]){
Multi3 m1=new Multi3();  Thread t1 =new Thread(m1);  t1.start();
}
}
```

Output:
thread is running...

3. Java program for sleep() methods

```
public class MyClass extends Thread
{
public static int amount = 0;
public static void main(String[] args)
{
MyClass thread = new MyClass();
thread.start();
while(thread.isAlive())
{ System.out.println("Waiting..."); }
System.out.println("Main: " + amount);
amount++;
System.out.println("Main: " + amount); }
public void run()
{ amount++; }
}
```

Output:

```
Waiting...
Waiting...
Waiting...
... (repeated multiple times)
Main: 1
Main: 2
```

4.Java program for sleep() methods

```
class TestSleepMethod1 extends Thread
{
public void run()
{
for(int i=1;i<5;i++)
{
try{Thread.sleep(500);}
catch(InterruptedException e){System.out.println(e);}  System.out.println(i);
}
}
public static void main(String args[])
{
TestSleepMethod1 t1=new TestSleepMethod1();  TestSleepMethod1 t2=new TestSleepMethod1();
t1.start();
t2.start(); } }
output:
1
2
1
2
3
3
4
4
```

5.java program for getName(),setName(String) and getId() method

```
class TestMethod3 extends Thread {
   public void run() {
      System.out.println("running...");
   }

   public static void main(String args[]) {
      TestMethod3 t1 = new TestMethod3();
      TestMethod3 t2 = new TestMethod3();

      System.out.println("Name of t1:" + t1.getName());
      System.out.println("Name of t2:" + t2.getName());

      System.out.println("id of t1:" + t1.getId());

      t1.start();
```

```
        t2.start();

        t1.setName("CBIT");
        System.out.println("After changing name of t1:" + t1.getName());
    }
}
```

Output:

```
Name of t1:Thread-0
Name of t2:Thread-1
id of t1:11
running...
running...
After changing name of t1:CBIT
```

6.Java program for thread priority methods

```
public class JavaSetPriorityExp1 extends Thread {
    public JavaSetPriorityExp1(String name) {
        super(name);
    }

    public void run() {
        System.out.println(getName() + " priority is: " + getPriority());
    }

    public static void main(String[] args) {
        // Thread with max priority (10)
        JavaSetPriorityExp1 t1 = new JavaSetPriorityExp1("Thread-1");
        t1.setPriority(Thread.MAX_PRIORITY);  // 10
        t1.start();

        // Thread with priority 4
        JavaSetPriorityExp1 t2 = new JavaSetPriorityExp1("Thread-2");
        t2.setPriority(4);
        t2.start();
    }
}
Thread-1 priority is: 10
Thread-2 priority is: 4
```

7 Java program without Synchronization

```
class Table{
void printTable(int n)
{
//method not synchronized
for(int i=1;i<=5;i++)
{
System.out.println(n*i);
```

```
try{
Thread.sleep(400);
}catch(Exception e){System.out.println(e);}       }
}}
class MyThread2 extends Thread
{
Table t;  MyThread2(Table t)
{
this.t=t;
}
public void run()
{ t.printTable(100);
} }
class MyThread1 extends Thread
{
Table t;  MyThread1(Table t){  this.t=t;
}
public void run(){
t.printTable(5);
} }
class TestSynchronization1{
public static void main(String args[]){  Table obj = new Table();//only one object
MyThread1 t1=new MyThread1(obj);  MyThread2 t2=new MyThread2(obj);  t1.start();
t2.start();
}
}

5
100
10
200
15
300
20
400
25
500
```

8 Java program with synchronization

```
class Table{
void printTable(int n){  synchronized(this){//synchronized block
for(int i=1;i<=5;i++){  System.out.println(n*i);  try{
Thread.sleep(400);
}catch(Exception e){System.out.println(e);}}          }          } }
class MyThread2 extends Thread
{
```

```
Table t;  MyThread2(Table t)
{
this.t=t;
}
public void run()
{ t.printTable(100);
} }
class MyThread1 extends Thread
{
Table t;  MyThread1(Table t){  this.t=t;
}
public void run(){
t.printTable(5);
} }
class TestSynchronization1{
public static void main(String args[]){  Table obj = new Table();//only one object
MyThread1 t1=new MyThread1(obj);  MyThread2 t2=new MyThread2(obj);  t1.start();
t2.start();
}
}
```

Output:

```
5
10
15
20
25
100
200
300
400
500
```

9.Java program for start_suspend_resume methods

```
public class JavaResumeExp extends Thread {
   public void run() {
      for (int i = 1; i < 3; i++) {
         try {
            sleep(500);
            System.out.println(Thread.currentThread().getName());
         } catch (InterruptedException e) {
            System.out.println(e);
         }
         System.out.println(i);
      }
   }
```

```
   public static void main(String args[]) {
      JavaResumeExp t1 = new JavaResumeExp();
      JavaResumeExp t2 = new JavaResumeExp();
      JavaResumeExp t3 = new JavaResumeExp();

      t1.start();
      t2.start();
      t2.suspend();
      t3.start();
      t2.resume();
   }
}
```

Output: Thread-0
1
Thread-0
2
Thread-2
1
Thread-2
2
Thread-1
1
Thread-1
2

10 .Java program for extending Thread class

```
class MyThread extends Thread
{
public void run()
{  for(int i =0;i<10;i++){System.out.println(“child Thread”);}}}

class ThreadDemo
{
   public static void main(String[] args)
   {
      MyThread t = new MyThread(); // Instantiation of a Thread
      t.start(); // starting of a Thread

      for(int i = 0; i < 5; i++)
      {
         System.out.println("main thread");
      }
   }
}
```

child Thread

main thread
child Thread
main thread
child Thread
main thread
child Thread
main thread
child Thread
main thread
child Thread
child Thread
child Thread
child Thread
child Thread

11**Java program for implementing Runnable interface**

```
Class MyRunnable implements Runnable
{
public void run()
{  for(int I =0;i<10;i++){System.out.println(“child Thread”);}}}

class ThreadDemo
{
   public static void main(String[] args)
   {
      MyRunnable r = new MyRunnable();
      Thread t = new Thread(r); // here r is a Target Runnable
      t.start();

      for (int i = 0; i < 10; i++)
      {
         System.out.println("main thread");
      }
   }
}
```

main thread
child thread
main thread
child thread
main thread
child thread
child thread
main thread
main thread
child thread
child thread
main thread
child thread
main thread

```
main thread
child thread
child thread
child thread
child thread
main thread
```

12. Java program for getname() , setname() methods

```
class MyThread extends Thread
{
}

class ThreadDemo
{
   public static void main(String args[])
   {
      System.out.println(Thread.currentThread().getName()); // main
      MyThread t = new MyThread();
      System.out.println(t.getName()); // Thread-0
Thread.currentThread().setName("itl thread");
      System.out.println(Thread.currentThread().getName()); // itl-thread
   }
}
main
Thread-0
itl thread
```

13.Java program for yield() method

```
class MyThread extends Thread
{
   public void run()
   {
      for (int i = 0; i < 5; i++)
      {
         Thread.yield();
         System.out.println("child thread");
      }
   }
}

class ThreadYieldDemo
{
   public static void main(String[] args)
   {
      MyThread t = new MyThread();
      t.start();
      for (int i = 0; i < 5; i++)
      {
         System.out.println("main thread");
      }
```

```
    }
}
```

```
main thread
child thread
main thread
child thread
main thread
child thread
main thread
child thread
main thread
child thread
```

14.Java program for yield() method – producer – consumer programme.

```
public class YieldExample {
    public static void main(String[] args) {
        Thread producer = new Thread(new ProducerTask(), "Producer");
        Thread consumer = new Thread(new ConsumerTask(), "Consumer");
        producer.start();
        consumer.start();
    }
}

class ProducerTask implements Runnable {
    @Override
    public void run() {
        for (int i = 0; i < 5; i++) {
            System.out.println(Thread.currentThread().getName() + ": Producing item " + i);
            Thread.yield();       }
    }
}
class ConsumerTask implements Runnable {
    @Override
    public void run() {
        for (int i = 0; i < 5; i++) {
            System.out.println(Thread.currentThread().getName() + ": Consuming item " + i);
        }
    }
}
```

```
Producer: Producing item 0
Consumer: Consuming item 0
Producer: Producing item 1
Consumer: Consuming item 1
Producer: Producing item 2
Consumer: Consuming item 2
Producer: Producing item 3
Consumer: Consuming item 3
Producer: Producing item 4
Consumer: Consuming item 4
```

15.Java program for join() methods

```
class MyThread extends Thread {
   private String threadName;

   MyThread(String name) {
      threadName = name;
   }

   public void run() {
      System.out.println(threadName + " is starting.");
      try {
         // Simulate some work with thread sleep
         Thread.sleep(2000);
      } catch (InterruptedException e) {
         System.out.println(threadName + " interrupted.");
      }
      System.out.println(threadName + " is finished.");
   }
}

public class JoinExample {
   public static void main(String[] args) {
      // Create threads
      MyThread t1 = new MyThread("Thread-1");
      MyThread t2 = new MyThread("Thread-2");
      MyThread t3 = new MyThread("Thread-3");

      // Start threads
      t1.start();
      t2.start();
      t3.start();

      // Use join() method to ensure main thread waits for t1, t2, and t3 to finish
      try {
         System.out.println("Waiting for threads to finish.");
         t1.join();  // Wait for Thread-1 to finish
         t2.join();  // Wait for Thread-2 to finish
         t3.join();  // Wait for Thread-3 to finish
      } catch (InterruptedException e) {
         System.out.println("Main thread interrupted.");
      }

      System.out.println("All threads have finished execution.");
   }
}
```

```
Thread-1 is starting.
Thread-2 is starting.
Thread-3 is starting.
Waiting for threads to finish.
Thread-1 is finished.
Thread-2 is finished.
Thread-3 is finished.
All threads have finished execution.
```

16.Java program sleep methods

```
class MyThread extends Thread {
   public void run() {
      try {
         for (int i = 0; i < 5; i++) {
            System.out.println("i am lazy Thread :" + i);
            Thread.sleep(2000);
         }
      } catch (InterruptedException e) {
         System.out.println("i got interrupted");
      }
   }
}

class ThreadInterruptDemo {
   public static void main(String[] args) {
      MyThread t = new MyThread();
      t.start();
      // t.interrupt(); //--->1
      System.out.println("end of main thread");
   }
}
```

```
end of main thread
i am lazy Thread :0
(wait 2 seconds)
i am lazy Thread :1
(wait 2 seconds)
i am lazy Thread :2
(wait 2 seconds)
i am lazy Thread :3
(wait 2 seconds)
i am lazy Thread :4
```

17. Java program for synchronized , sleep methods

```
class Display {
   public synchronized void wish(String name) {
      for (int i = 0; i < 5; i++) {
         System.out.print("good morning:");
         try {
            Thread.sleep(1000);
         } catch (InterruptedException e) {
         }
         System.out.println(name);
      }
   }
}

class MyThread extends Thread {
   Display d;
   String name;

   MyThread(Display d, String name) {
      this.d = d;
      this.name = name;
   }

   public void run() {
      d.wish(name);
   }
}

class SynchronizedDemo {
   public static void main(String[] args) {
      Display d1 = new Display();
      MyThread t1 = new MyThread(d1, "CBIT");
      MyThread t2 = new MyThread(d1, "CSM");
      t1.start();
      t2.start();
   }
}
```

Output with synchronizing wish method

```
good morning:CBIT
good morning:CSM
good morning:CBIT
good morning:CSM
good morning:CBIT
good morning:CSM
good morning:CBIT
good morning:CSM
```

good morning:CBIT
good morning:CSM

Output with synchronization of wish method

good morning:CBIT
good morning:CBIT
good morning:CBIT
good morning:CBIT
good morning:CBIT
good morning:CSM
good morning:CSM
good morning:CSM
good morning:CSM
good morning:CSM

18.Java program for Wait(), Notify() methods

```
class ThreadA
{
   public static void main(String[] args) throws InterruptedException
   {
      ThreadB b = new ThreadB();
      b.start();
      synchronized(b)
      {
         System.out.println("main Thread calling wait() method!");//step-1
         b.wait();
         System.out.println("main Thread got notification call!");//step-4
         System.out.println(b.total);
      }
   }
}

class ThreadB extends Thread
{
   int total = 0;
   public void run()
   {
      synchronized(this)
      {
         System.out.println("child thread starts calculation");//step-2
         for(int i = 0; i <= 100; i++)
         {
            total = total + i;
         }
         System.out.println("child thread giving notification call");//step-3
         this.notify();
      }
```

```
    }
}
```

Output:
main Thread calling wait() method! // step-1
child thread starts calculation   // step-2
child thread giving notification call // step-3
main Thread got notification call!   // step-4

WEEK 9: Implement the collection framework classes with Iterator/List Iterator (lab date : 13.10.25 - programs)

1. Collection Interface Example

```
import java.util.*;

public class CollectionInterfaceDemo {
   public static void main(String[] args) {
      Collection<String> fruits = new ArrayList<>();
      fruits.add("Apple");
      fruits.add("Banana");
      fruits.add("Cherry");

      System.out.println("Fruits Collection: " + fruits);
      System.out.println("Contains Banana? " + fruits.contains("Banana"));
      fruits.remove("Cherry");
      System.out.println("After removal: " + fruits);
   }
}
**Output:**
Fruits Collection: [Apple, Banana, Cherry]
Contains Banana? true
After removal: [Apple, Banana]
```

2.Set Interface Example

```
import java.util.*;

public class SetDemo {
   public static void main(String[] args) {
      Set<String> set = new HashSet<>();
      set.add("Red");
      set.add("Blue");
      set.add("Green");
      set.add("Red");

      System.out.println("HashSet Elements: " + set);
   }
}
```

```

**Output:**

HashSet Elements: [Red, Blue, Green] (Order may vary)

2. List Interface Example

```
import java.util.*;

public class ListDemo {
   public static void main(String[] args) {
      List<String> list = new ArrayList<>();
      list.add("A");
      list.add("B");
      list.add("C");
      list.add(1, "Inserted");

      System.out.println("ArrayList Elements: " + list);
      System.out.println("Element at index 2: " + list.get(2));
   }
}
```

**Output:**

ArrayList Elements: [A, Inserted, B, C]
Element at index 2: B

4.Map Interface Example

```
import java.util.*;

public class MapDemo {
   public static void main(String[] args) {
      Map<Integer, String> map = new HashMap<>();
      map.put(1, "One");
      map.put(2, "Two");
      map.put(3, "Three");

      System.out.println("HashMap: " + map);
      System.out.println("Value of key 2: " + map.get(2));
   }
}
```

Output:
HashMap: {1=One, 2=Two, 3=Three}
Value of key 2: Two

5. ArrayList Example

```
import java.util.*;

public class ArrayListDemo {
```
```

```
    public static void main(String[] args) {
        ArrayList<Integer> numbers = new ArrayList<>();
        numbers.add(10);
        numbers.add(20);
        numbers.add(30);
        System.out.println("ArrayList: " + numbers);
    }
}
```
Output:
ArrayList: [10, 20, 30]

6. LinkedList Example
```
import java.util.*;

public class LinkedListDemo {
    public static void main(String[] args) {
        LinkedList<String> animals = new LinkedList<>();
        animals.add("Dog");
        animals.add("Cat");
        animals.addFirst("Elephant");
        animals.addLast("Horse");
        System.out.println("LinkedList: " + animals);
    }
}
```

Output:
LinkedList: [Elephant, Dog, Cat, Horse]

7. HashSet Example
```
import java.util.*;

public class HashSetDemo {
    public static void main(String[] args) {
        HashSet<String> names = new HashSet<>();
        names.add("Ravi");
        names.add("Anita");
        names.add("Ravi");
        System.out.println("HashSet: " + names);
    }
}
```
Output:
HashSet: [Anita, Ravi] (Order may vary)

8 TreeSet Example

```
import java.util.*;

public class TreeSetDemo {
    public static void main(String[] args) {
        TreeSet<Integer> treeSet = new TreeSet<>();
```

```
        treeSet.add(40);
        treeSet.add(10);
        treeSet.add(30);
        System.out.println("TreeSet (Sorted): " + treeSet);
    }
}
```
Output:

```
TreeSet (Sorted): [10, 30, 40]
```

9 HashMap Example

```
import java.util.*;

public class HashMapDemo {
    public static void main(String[] args) {
        HashMap<Integer, String> map = new HashMap<>();
        map.put(101, "John");
        map.put(102, "Mike");
        map.put(103, "Sara");

        System.out.println("HashMap Elements: " + map);
        System.out.println("Keys: " + map.keySet());
        System.out.println("Values: " + map.values());
    }
}
```
Output:

```
HashMap Elements: {101=John, 102=Mike, 103=Sara}
Keys: [101, 102, 103]
Values: [John, Mike, Sara]
```

10. TreeMap Example

```
import java.util.*;

public class TreeMapDemo {
    public static void main(String[] args) {
        TreeMap<Integer, String> map = new TreeMap<>();
        map.put(3, "Three");
        map.put(1, "One");
        map.put(2, "Two");

        System.out.println("TreeMap (Sorted): " + map);
    }
}
```
Output:

```
TreeMap (Sorted): {1=One, 2=Two, 3=Three}
```

11 Iterator Example

```
import java.util.*;

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

```
        List<String> list = Arrays.asList("Apple", "Banana", "Cherry");
        Iterator<String> itr = list.iterator();

        System.out.println("Iterating with Iterator:");
        while (itr.hasNext()) {
            System.out.println(itr.next());
        }
    }
}
```
Output:
```
Iterating with Iterator:
Apple
Banana
Cherry
```

12 ListIterator Example

```
import java.util.*;

public class ListIteratorDemo {
    public static void main(String[] args) {
        List<String> list = new ArrayList<>(Arrays.asList("A", "B", "C"));
        ListIterator<String> litr = list.listIterator();

        System.out.println("Forward Iteration:");
        while (litr.hasNext()) {
            System.out.println(litr.next());
        }

        System.out.println("Backward Iteration:");
        while (litr.hasPrevious()) {
            System.out.println(litr.previous());
        }
    }
}
```

Output:
```
Forward Iteration:
A
B
C
Backward Iteration:
C
B
A
```

13 Comparable Interface Example

```
import java.util.*;

class Student implements Comparable<Student> {
```

```
    int id;
    String name;

    Student(int id, String name) {
        this.id = id;
        this.name = name;
    }

    public int compareTo(Student s) {
        return this.id - s.id;
    }

    public String toString() {
        return id + " - " + name;
    }
}

public class ComparableDemo {
    public static void main(String[] args) {
        List<Student> list = new ArrayList<>();
        list.add(new Student(3, "John"));
        list.add(new Student(1, "Asha"));
        list.add(new Student(2, "Ravi"));

        Collections.sort(list);
        System.out.println("Sorted by ID: " + list);
    }
}
```

Output:
`Sorted by ID: [1 - Asha, 2 - Ravi, 3 - John]

14 Comparator Interface Example

```
import java.util.*;

class Person {
    String name;
    int age;

    Person(String name, int age) {
        this.name = name;
        this.age = age;
    }

    public String toString() {
        return name + " (" + age + ")";
    }
}

public class ComparatorDemo {
```

```
    public static void main(String[] args) {
        List<Person> list = new ArrayList<>();
        list.add(new Person("Kiran", 30));
        list.add(new Person("Anu", 25));
        list.add(new Person("Vishal", 28));

        Collections.sort(list, (p1, p2) -> p1.age - p2.age);
        System.out.println("Sorted by Age: " + list);
    }
}
```

Output:
Sorted by Age: [Anu (25), Vishal (28), Kiran (30)]

15 Functional Interface & Lambda Example

```
@FunctionalInterface
interface MessagePrinter {
    void print(String msg);
}

public class LambdaDemo {
    public static void main(String[] args) {
        MessagePrinter printer = (msg) -> System.out.println("Message: " + msg);
        printer.print("Hello from Lambda Expression!");
    }
}
```

Output:
Message: Hello from Lambda Expression!

16 Java 8 Stream and Lambda Example

```
import java.util.*;

public class StreamDemo {
    public static void main(String[] args) {
        List<String> names = Arrays.asList("Anu", "Kiran", "Arun", "Vishal");

        System.out.println("Names starting with 'A':");
        names.stream()
             .filter(n -> n.startsWith("A"))
             .forEach(System.out::println);
    }
}
```

Output:
Names starting with 'A':
Anu
Arun

WEEK 10: java programs on Files , collections (lab date : 20.10.25 & 22.10.25 - 21 programs)

1.program on LinkedList

```
import java.util.*;
class LinkedListDemo
{
   public static void main(String[] args)
   {
      LinkedList l = new LinkedList();
      l.add("ashok");
      l.add(30);
      l.add(null);
      l.add("ashok");
      System.out.println(l); //[ashok, 30, null, ashok]

      l.set(0, "software");
      System.out.println(l); //[software, 30, null, ashok]

      l.set(0, "venky");
      System.out.println(l); //[venky, 30, null, ashok]

      l.removeLast();
      System.out.println(l); //[venky, 30, null]

      l.addFirst("vvv");
      System.out.println(l); //[vvv, venky, 30, null]
   }
}
```

Output:

[ashok, 30, null, ashok] — after adding all elements.
[software, 30, null, ashok] — replaced first element with "software".
[venky, 30, null, ashok] — replaced first element again.
[venky, 30, null] — last element "ashok" removed.
[vvv, venky, 30, null] — "vvv" added to the beginning.

2. compartTo()

```
class Test{
public static void main(String[] args)    {
System.out.println("A".compareTo("Z"));   // -25
System.out.println("Z".compareTo("K"));   // 15
 System.out.println("A".compareTo("A"));   // 0
  // System.out.println("A".compareTo(new Integer(10)));
  // Test.java:8: compareTo(java.lang.String) in java.lang.String cannot be applied to
(java.lang.Integer)

  // System.out.println("A".compareTo(null)); // NullPointerException
   }
}
```

Output:

- "A".compareTo("Z") → -25 (because 'A' < 'Z' by 25 in Unicode value).

- "Z".compareTo("K") → 15 (because 'Z' > 'K' by 15 in Unicode value).
- "A".compareTo("A") → 0 (both are equal).
- "A".compareTo(new Integer(10)) → **Compile-time error**, since compareTo expects a String.
- "A".compareTo(null) → **NullPointerException**, because you're comparing with null.

3. Vector program & Enumeration

```
import java.util.Vector;
import java.util.Enumeration;

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

        // Create a Vector
        Vector<String> fruits = new Vector<>();

        // Add elements to the Vector
        fruits.add("Apple");
        fruits.add("Banana");
        fruits.add("Mango");
        fruits.add("Grapes");
        fruits.add("Orange");

        // Display the Vector
        System.out.println("Fruits in the Vector: " + fruits);

        // Get Enumeration object from the Vector
        Enumeration<String> fruitEnum = fruits.elements();

        // Iterate through the elements using Enumeration
        System.out.println("\nIterating using Enumeration:");
        while (fruitEnum.hasMoreElements()) {
            String fruit = fruitEnum.nextElement();
            System.out.println(fruit);
        }
    }
}
```

Output:
Fruits in the Vector: [Apple, Banana, Mango, Grapes, Orange]

Iterating using Enumeration:
Apple
Banana
Mango
Grapes
Orange

4.ArrayList and Iterator

```
import java.util.ArrayList;
import java.util.Iterator;

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

```
        // Create an ArrayList of Strings
        ArrayList<String> cities = new ArrayList<>();

        // Add elements to the ArrayList
        cities.add("New York");
        cities.add("London");
        cities.add("Tokyo");
        cities.add("Sydney");
        cities.add("Paris");

        // Display the ArrayList
        System.out.println("Cities in the ArrayList: " + cities);

        // Get an Iterator object
        Iterator<String> cityIterator = cities.iterator();

        // Iterate through the elements using Iterator
        System.out.println("\nIterating using Iterator:");
        while (cityIterator.hasNext()) {
            String city = cityIterator.next();
            System.out.println(city);
        }

        // Example of removing an element using Iterator
        cityIterator = cities.iterator(); // Re-initialize iterator
        while (cityIterator.hasNext()) {
            String city = cityIterator.next();
            if (city.equals("Tokyo")) {
                cityIterator.remove();  // Safe removal using iterator
            }
        }

        // Display updated list
        System.out.println("\nArrayList after removing 'Tokyo': " + cities);
    }
}
```

Output:

```
Cities in the ArrayList: [New York, London, Tokyo, Sydney, Paris]

Iterating using Iterator:
New York
London
Tokyo
Sydney
Paris

ArrayList after removing 'Tokyo': [New York, London, Sydney, Paris]
```

5.LinkedList and ListIterator

```
import java.util.LinkedList;
import java.util.ListIterator;
```

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

        // Create a LinkedList of Strings
        LinkedList<String> animals = new LinkedList<>();

        // Add elements to the LinkedList
        animals.add("Dog");
        animals.add("Cat");
        animals.add("Elephant");
        animals.add("Tiger");
        animals.add("Lion");

        System.out.println("Original LinkedList: " + animals);

        // Get a ListIterator
        ListIterator<String> iterator = animals.listIterator();

        // Traverse forward using ListIterator
        System.out.println("\nTraversing forward:");
        while (iterator.hasNext()) {
            System.out.println("Index " + iterator.nextIndex() + " → " + iterator.next());
        }

        // Traverse backward using ListIterator
        System.out.println("\nTraversing backward:");
        while (iterator.hasPrevious()) {
            System.out.println("Index " + iterator.previousIndex() + " → " + iterator.previous());
        }

        // Modify elements using ListIterator
        iterator = animals.listIterator(); // reset iterator to start
        while (iterator.hasNext()) {
            String animal = iterator.next();
            if (animal.equals("Cat")) {
                iterator.set("Kitten"); // replace 'Cat' with 'Kitten'
            } else if (animal.equals("Elephant")) {
                iterator.add("Giraffe"); // add a new element after 'Elephant'
            }
        }

        System.out.println("\nLinkedList after modifications: " + animals);

        // Demonstrate removing an element using ListIterator
        iterator = animals.listIterator();
        while (iterator.hasNext()) {
            String animal = iterator.next();
            if (animal.equals("Tiger")) {
                iterator.remove(); // remove 'Tiger' safely
            }
        }

        System.out.println("\nLinkedList after removing 'Tiger': " + animals);
    }
```

```
}
```
Output:
Original LinkedList: [Dog, Cat, Elephant, Tiger, Lion]

Traversing forward:
Index 0 → Dog
Index 1 → Cat
Index 2 → Elephant
Index 3 → Tiger
Index 4 → Lion

Traversing backward:
Index 4 → Lion
Index 3 → Tiger
Index 2 → Elephant
Index 1 → Cat
Index 0 → Dog

LinkedList after modifications: [Dog, Kitten, Elephant, Giraffe, Tiger, Lion]

LinkedList after removing 'Tiger': [Dog, Kitten, Elephant, Giraffe, Lion]

6. File – exist(),mkdir(),createnewfile()

```
import java.io.File;
import java.io.IOException;

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

        // Define directory and file paths
        String dirName = "MyFolder";
        String fileName = "example.txt";

        // Create a File object for the directory
        File directory = new File(dirName);

        // Check if the directory exists
        if (!directory.exists()) {
            // Create a new directory
            boolean dirCreated = directory.mkdir();
            if (dirCreated) {
                System.out.println("Directory created: " + directory.getAbsolutePath());
            } else {
                System.out.println("Failed to create directory.");
            }
        } else {
            System.out.println("Directory already exists: " + directory.getAbsolutePath());
        }

        // Create a File object for the file inside the directory
        File file = new File(directory, fileName);
```

```
        try {
            // Check if file exists
            if (!file.exists()) {
                // Create a new file
                boolean fileCreated = file.createNewFile();
                if (fileCreated) {
                    System.out.println("File created: " + file.getAbsolutePath());
                } else {
                    System.out.println("Failed to create file.");
                }
            } else {
                System.out.println("File already exists: " + file.getAbsolutePath());
            }

        } catch (IOException e) {
            System.out.println("An error occurred while creating the file.");
            e.printStackTrace();
        }
    }
}
```

Output 1:

Directory created: C:\Users\YourName\MyFolder
File created: C:\Users\YourName\MyFolder\example.txt

Output 2:
Directory already exists: C:\Users\YourName\MyFolder
File already exists: C:\Users\YourName\MyFolder\example.txt

7. Read a single byte from file using InputStream

```
import java.io.FileInputStream;
import java.io.IOException;

public class ReadSingleByte {
    public static void main(String[] args) {
        try (FileInputStream fis = new FileInputStream("input1.txt")) {
            int data = fis.read();
            System.out.println("First byte read: " + (char) data);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Output:

Input File: input1.txt → Hello

Output:

First byte read: H

8. Read all bytes using InputStream

```
import java.io.FileInputStream;
import java.io.IOException;

public class ReadAllBytes {
    public static void main(String[] args) {
        try (FileInputStream fis = new FileInputStream("input2.txt")) {
            int data;
            while ((data = fis.read()) != -1) {
                System.out.print((char) data);
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Output:

Input File: input2.txt → Java IO Example

Output:

Java IO Example

9. Write bytes using OutputStream

```
import java.io.FileOutputStream;
import java.io.IOException;

public class WriteBytes {
    public static void main(String[] args) {
        try (FileOutputStream fos = new FileOutputStream("output1.txt")) {
            String str = "Writing with OutputStream";
            fos.write(str.getBytes());
            System.out.println("Data written successfully!");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Output:

Output File (output1.txt): Writing with OutputStream

10. Copy file content using InputStream and OutputStream

```
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;

public class CopyFile {
    public static void main(String[] args) {
        try (FileInputStream fis = new FileInputStream("source.txt");
             FileOutputStream fos = new FileOutputStream("dest.txt")) {
            int data;
            while ((data = fis.read()) != -1) {
                fos.write(data);
            }
```

```
            System.out.println("File copied successfully!");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Output:

Output: File copied successfully!

11. Using Reader to read characters

```
import java.io.FileReader;
import java.io.IOException;

public class ReaderExample {
    public static void main(String[] args) {
        try (FileReader reader = new FileReader("reader1.txt")) {
            int ch;
            while ((ch = reader.read()) != -1) {
                System.out.print((char) ch);
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Output:

Reads characters from reader1.txt

12. Using Writer to write characters

```
import java.io.FileWriter;
import java.io.IOException;

public class WriterExample {
    public static void main(String[] args) {
        try (FileWriter writer = new FileWriter("writer1.txt")) {
            writer.write("This is written using Writer class.");
            System.out.println("Data written successfully!");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Output:

Output File (writer1.txt): This is written using Writer class.

13. Append text using FileWriter

```
import java.io.FileWriter;
import java.io.IOException;

public class AppendFileWriter {
    public static void main(String[] args) {
        try (FileWriter fw = new FileWriter("writer1.txt", true)) {
```

```
            fw.write("\nAppending new line.");
            System.out.println("Text appended!");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Output:

Appends text to writer1.txt

14. Read text line by line using BufferedReader

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;

public class BufferedReaderExample {
    public static void main(String[] args) {
        try (BufferedReader br = new BufferedReader(new
FileReader("buffered1.txt"))) {
            String line;
            while ((line = br.readLine()) != null) {
                System.out.println("Line: " + line);
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Output:

Reads lines from buffered1.txt

15. Write text using BufferedWriter

```
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;

public class BufferedWriterExample {
    public static void main(String[] args) {
        try (BufferedWriter bw = new BufferedWriter(new
FileWriter("buffered2.txt"))) {
            bw.write("This is written using BufferedWriter.");
            bw.newLine();
            bw.write("Second line here.");
            System.out.println("Data written successfully!");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Output:

Output File (buffered2.txt):
This is written using BufferedWriter.
Second line here.

16. Count number of characters in a file

```
import java.io.FileReader;
import java.io.IOException;

public class CharCount {
    public static void main(String[] args) {
        int count = 0;
        try (FileReader fr = new FileReader("count.txt")) {
            while (fr.read() != -1) {
                count++;
            }
            System.out.println("Total characters: " + count);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Output:

Output Example:
Total characters: 24

17. Count number of lines using BufferedReader

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;

public class LineCount {
    public static void main(String[] args) {
        int lineCount = 0;
        try (BufferedReader br = new BufferedReader(new FileReader("lines.txt")))
{
            while (br.readLine() != null) {
                lineCount++;
            }
            System.out.println("Total lines: " + lineCount);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Output:

Output Example:
Total lines: 3

18. Read from keyboard using InputStream

```
import java.io.IOException;
```

```
public class KeyboardRead {
    public static void main(String[] args) throws IOException {
        System.out.print("Enter a character: ");
        int ch = System.in.read();
        System.out.println("You entered: " + (char) ch);
    }
}
```

Output:

Input: A

Output: You entered: A

19. Write to console using OutputStream

```
import java.io.IOException;

public class ConsoleWrite {
    public static void main(String[] args) throws IOException {
        String msg = "Hello from OutputStream!";
        System.out.write(msg.getBytes());
        System.out.flush();
    }
}
```

Output:

Output:

Hello from OutputStream!

20. Copy content using BufferedReader and BufferedWriter

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;

public class CopyBuffered {
    public static void main(String[] args) {
        try (BufferedReader br = new BufferedReader(new
FileReader("source2.txt"));
             BufferedWriter bw = new BufferedWriter(new FileWriter("dest2.txt")))
{
            String line;
            while ((line = br.readLine()) != null) {
                bw.write(line);
                bw.newLine();
            }
            System.out.println("File copied using buffering!");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Output:

Output:

File copied using buffering!

21. Convert lowercase to uppercase while copying

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;

public class UpperCaseCopy {
    public static void main(String[] args) {
        try (BufferedReader br = new BufferedReader(new
FileReader("inputcase.txt"));
             BufferedWriter bw = new BufferedWriter(new
FileWriter("outputcase.txt"))) {
            String line;
            while ((line = br.readLine()) != null) {
                bw.write(line.toUpperCase());
                bw.newLine();
            }
            System.out.println("Conversion to uppercase completed!");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Output:

Output File (outputcase.txt):

ORIGINAL TEXT CONVERTED TO UPPERCASE