Name: Amit Kumar Parhi

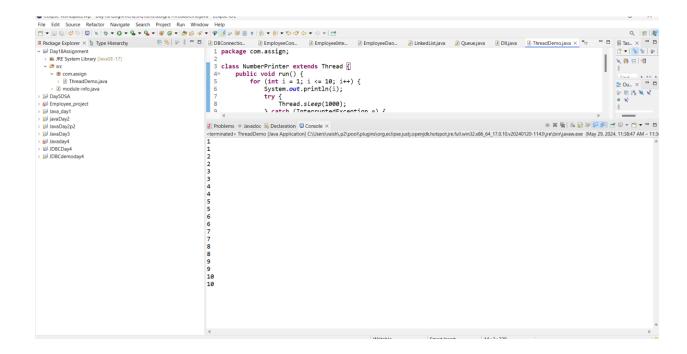
Email:amitkparhi07@gmail.com

Day 18:

Task 1: Creating and Managing Threads Write a program that starts two threads, where each thread prints numbers from 1 to 10 with a 1-second delay between each number

```
package com.assign;
class NumberPrinter extends Thread {
    public void run() {
        for (int i = 1; i <= 10; i++) {
            System.out.println(i);
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
public class ThreadDemo {
    public static void main(String[] args) {
        Thread thread1 = new NumberPrinter();
        Thread thread2 = new NumberPrinter();
        thread1.start();
        thread2.start();
    }
}
```

Output:



Task 2: States and Transitions Create a Java class that simulates a thread going through different lifecycle states: NEW, RUNNABLE, WAITING, TIMED_WAITING, BLOCKED, and TERMINATED. Use methods like sleep(), wait(), notify(), and join() to demonstrate these states..

```
package com.assign;
public class Task2 {
    private static final Object Lock = new Object();
    public static void main(String[] args) {
        Thread thread = new Thread(new RunnableTask());
        System.out.println("Thread state after creation: " +
thread.getState());
        thread.start();
        System.out.println("Thread state after calling start(): " +
thread.getState());
        try {
            Thread.sleep(100);
            synchronized (Lock) {
                Lock.notify();
            }
            Thread.sleep(200);
            System.out.println("Thread state during sleep(): " +
thread.getState());
            synchronized (Lock) {
```

```
System.out.println("Thread state when trying to
acquire lock: " + thread.getState());
            thread.join();
          System.out.println("Thread state after termination: " +
thread.getState());
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
    private static class RunnableTask implements Runnable {
        @Override
        public void run() {
            synchronized (Lock) {
                try {
                    Lock.wait();
                    System.out.println("Thread state in wait(): " +
Thread.currentThread().getState());
                    Thread.sleep(100);
                    System.out.println("Thread state in timed wait: "
+ Thread.currentThread().getState());
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
            try {
                Thread.sleep(100);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
   }
}
```

Output:

```
1 package com.assign;
        > al. JRE System Library [JavaSE-17]

v  src

v  com.assign
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         × 11
                                                                                                                                                                                                                                      3 public class Task2 {
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             Find

    BankDemo.java
    CommunicationDemo.java
    Task2.java

                                                                                                                                                                                                                                                               private static final Object Lock = new Object();
                                                                                                                                                                                                                                                               public static void main(String[] args) {
    Thread thread = new Thread(new RunnableTask());
                                   Task5.java
                                                                                                                                                                                                                        > III Taskfi.iava

    I Task7,java
    ThreadDemo.java
    ThreadStateAndTransition.java
                    > D module-info.java
> [2] module-info,java

> DAY19Assign

> M. RE System Library [JavaSE-17]

> Ser src

- (default package)

> (default package)

| (default package)
| (default package)
| (default package)
| (default package)
| (default 
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         ₽ Ou
       ## DaySOSA

## Employee_project

## Lava_day1

> ## JRE System Library [JavaSE-17]

> ## Com.wipro

> ## BuiltinAnno.java

> ## Completiturejava

## Completiturejava
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              customAnnoation.java

    Employee.java
    ExecutorExamp.java
    GenLBUB.java
    LinkedListDemo.java

    Manager.java
    MyGenClass.java
    Salary.java
```

Task 3: Synchronization and Inter-thread Communication Implement a producer-consumer problem using wait() and notify() methods to handle the correct processing sequence between threads.

```
package com.assign;
class Buffer {
    private int data;
    private boolean empty = true;
    public synchronized void produce(int value) {
        while (!empty) {
            try {
                wait();
            } catch (InterruptedException e) {
                e.printStackTrace();
        }
        data = value;
        empty = false;
        notify();
    }
    public synchronized int consume() {
        while (empty) {
            try {
                wait();
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
        empty = true;
        notify();
        return data;
}
```

```
class Producer extends Thread {
    private Buffer buffer;
    public Producer(Buffer buffer) {
        this.buffer = buffer;
    }
    public void run() {
        for (int i = 1; i <= 10; i++) {
            buffer.produce(i);
            System.out.println("Produced: " + i);
        }
    }
}
class Consumer extends Thread {
    private Buffer buffer;
    public Consumer(Buffer buffer) {
        this.buffer = buffer;
    }
    public void run() {
        for (int i = 1; i <= 10; i++) {
            int value = buffer.consume();
            System.out.println("Consumed: " + value);
        }
    }
}
public class CommunicationDemo {
    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:

```
The sadd Source Selector Rangule Seaton. Project Sun. Window Help

If Package Displace X is Type Histority

If Package Displace X is Type Histority

If Description Research R
```

Task 4: Synchronized Blocks and Methods Write a program that simulates a bank account being accessed by multiple threads to perform deposits and withdrawals using synchronized methods to prevent race conditions.

```
package com.assign;
class BankAccount {
    private int balance = 0;
    public synchronized void deposit(int amount) {
        balance += amount;
        System.out.println("Deposited: " + amount);
    }
    public synchronized void withdraw(int amount) {
        if (balance >= amount) {
            balance -= amount;
            System.out.println("Withdrawn: " + amount);
        } else {
            System.out.println("Insufficient funds!");
        }
    }
}
class Transaction extends Thread {
    private BankAccount account;
    private boolean isDeposit;
    private int amount;
    public Transaction(BankAccount account, boolean isDeposit, int
amount) {
        this.account = account;
        this.isDeposit = isDeposit;
        this.amount = amount;
    }
    public void run() {
        if (isDeposit) {
            account.deposit(amount);
        } else {
            account.withdraw(amount);
        }
    }
```

```
}
public class BankDemo {
    public static void main(String[] args) {
        BankAccount account = new BankAccount();
        Transaction[] transactions = new Transaction[5];
        for (int i = 0; i < transactions.length; i++) {</pre>
            if (i % 2 == 0) {
                transactions[i] = new Transaction(account, true, 100);
            } else {
                transactions[i] = new Transaction(account, false, 50);
        }
        for (Transaction transaction : transactions) {
            transaction.start();
        }
    }
}
```

```
🕏 Package Explorer X 🔭 Type Hierarchy 🕒 😌 👺 🖟 📅 🗇 🗇 DBConnectio... 🚇 LinkedListjava 🚇 Queuejava 👚 DILjava 👚 ThreadDemojava 🕮 ThreadDetaijava 🕮 ThreadDetaijava 🗡 25 🚾 🗖
                                                           1 package com.assign;
> mt JRE System Library [JavaSE-17]

> the src

the com.assign
                                                             class BankAccount {
   private int balance = 0;

    BankDemo.java
    CommunicationDemo.java
    ThreadDemo.java
                                                                 public synchronized void deposit(int amount) {
                                                                      System.out.println("Deposited: " + amount);
      > ② module-info.java

Signatura
                                                                  public synchronized void withdraw(int amount) {
   if (balance >= amount) {
      balance -= amount;
}
> # Employee_project
                                                          118
                                                          12
13
14
15
16
17
                                                                            System.out.println("Withdrawn: " + amount);
 System.out.println("Insufficient funds!");
 > 🎏 JDBCdemoday4
                                                          18
                                                        21 class Transaction extends Thread [ 22 private BankAccount account; 23 private boolean isDeposit.
                                                                  private int amount;
                                                                 public Transaction(BankAccount account, boolean isDeposit, int amount) {
                                                                       this.account = account;
                                                                                                                                                                           - X % | & 5 P P -
                                                         Problems @ Javadoc & Declaration Q Console X
                                                         -terminated-BankDemo [Java Application] C:\Users\vaish\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.10.v20240120-1143\jre\bin\javaw.exe (May 29, 2024,
                                                         Deposited: 100
                                                        Deposited: 100
Withdrawn: 50
Deposited: 100
                                                        Withdrawn: 50
```

Task 5: Thread Pools and Concurrency Utilities Create a fixed-size thread pool and submit multiple tasks that perform complex calculations or I/O operations and observe the execution.

```
package com.assign;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.TimeUnit;
class ComplexCalculation implements Runnable {
    private int taskId;
    public ComplexCalculation(int taskId) {
        this.taskId = taskId;
    }
    @Override
    public void run() {
        System.out.println("Task " + taskId + " is starting.");
        try {
            TimeUnit. SECONDS. sleep(2);
        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
        System.out.println("Task " + taskId + " is completed.");
    }
}
public class Task5 {
    public static void main(String[] args) {
        ExecutorService executorService =
Executors.newFixedThreadPool(4);
        for (int i = 1; i <= 10; i++) {
            executorService.submit(new ComplexCalculation(i));
        }
        executorService.shutdown();
        try {
```

```
if (!executorService.awaitTermination(60,
TimeUnit.SECONDS)) {
                                                     executorService.shutdownNow();
                          } catch (InterruptedException e) {
                                       executorService.shutdownNow();
                          }
             }
}
                                                                          4 import java.util.concurrent.Executors;
5 import java.util.concurrent.TimeUnit;
  ⇒ M JRE System Library [JavaSE-17]
  ∨ ∰ src
∨ ∰ com.assign
                                                                             class ComplexCalculation implements Runnable {
       private int taskId;
       > 1 Task5.iava
                                                                                  public ComplexCalculation(int taskId) {
                                                                       10⊝
       > ② ThreadDemo.java
> ② ThreadStateAndTransition.java
                                                                                                                                                                                                                       - X % | B a B
                                                                       Problems @ Javadoc □ Declaration □ Console ×
> 🗓 module-info.java
                                                                      <terminated > Task5 [Java Application] C\Users\vaish\p2\pool\plugins\org.eclipse.justj.openjdkhotspot.jre.full.win32x86_64_17.0.10v20240120-1143\jre\bin\javaw.exe (May 29, 2) [Task 4 is starting.
  > A JRE System Library [JavaSE-17]
  ✓ ﷺ src
✓ ﷺ com.wipro.algo
                                                                       Task 1 is starting.
Task 3 is starting.
                                                                      Task 3 is starting.
Task 2 is starting.
Task 4 is completed.
Task 5 is starting.
Task 3 is completed.
Task 6 is completed.
Task 6 is starting.
Task 1 is completed.
Task 7 is starting.
Task 8 is starting.
Task 6 is completed.
Task 6 is completed.
    > 🗓 LinearSearch.java

• # com.wipro.dll
       > 🗓 DII.java
       → BigOn.java

    BigOnsquare.java
    Bigoone.java

       > 

ExpoDemo.java
        > 🗓 Logndemo.java

→ 

B com.wipro.ll

        > 🕖 LinkedList.java
                                                                       Task 9 is starting.
Task 5 is completed.
     > # com.wipro.q
                                                                       Task 10 is starting.
Task 8 is completed.
Task 7 is completed.
    > 🗾 JumpSearch.java
    > 🔝 LinearSearch.java

• # com.wipro.sort
                                                                       Task 10 is completed.
Task 9 is completed.
        > D BubbleSort.iava
     > # com.wipro.st

    ✓ 
        <sup>B</sup> com.wipro.tree

            ✓ BinaryTree.java

    ✓ Immodule-info.java
```

Task 6: Executors, Concurrent Collections, CompletableFuture Use an ExecutorService to parallelize a task that calculates prime numbers up to a given number and then use CompletableFuture to write the results to a file asynchronously.

```
package com.assign;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.stream.IntStream;
public class Task6 {
  static boolean isPrime(int num) {
    if (num <= 1) return false;</pre>
    for (int i = 2; i * i <= num; i++) {</pre>
      if (num % i == 0) return false;
    return true;
 public static void main(String[] args) throws Exception {
       int maxNumber = 10;
       ExecutorService executor = Executors.newFixedThreadPool(4);
       System.out.println("Finding prime numbers up to " + maxNumber);
       CompletableFuture<StringBuilder> primeNumbers =
CompletableFuture.supplyAsync(() -> {
         StringBuilder result = new StringBuilder();
         IntStream.rangeClosed(2, maxNumber).filter(Task6::isPrime)
           .forEach(prime -> result.append(prime).append(" "));
         return result;
       }, executor);
       primeNumbers.thenAcceptAsync(result -> {
         try {
java.nio.file.Files.writeString(java.nio.file.Paths.get("d:/data/prime")
numbers.txt"), result.toString());
           System.out.println("Prime numbers written to file:
prime numbers.txt");
```

```
} catch (Exception e) {
                                                                                        e.printStackTrace();
                                                         }, executor);
                                                         executor.shutdown();
                                         }
}
                                                                                                                                                                                    static poolean isprime(int num) {

→ ## com.assign

                                                                                                                                                               11
12
13
14
15
16 }
                                                                                                                                                                                         if (num <= 1) return false;
for (int i = 2; i * i <= num; i++) {
   if (num % i == 0) return false;</pre>
               > A BankDemo.iava

    CommunicationDemo.java
    Task5.java
              > 🗓 Task6.java
            Task7.java
ThreadDemo.java
17
18°
19
20
21
22
23
24
25
26
27
28
29
30
31
                                                                                                                                                                                 public static void main(String[] args) throws Exception {
   int maxNumber = 10;
   ExecutorService executor = Executors.newFixedThreadPool(4);
                                                                                                                                                                                                 System.out.println("Finding prime numbers up to " + maxNumber);
→ Pair,java

⊕ com.wipro.assign19

→ M module-info.java

Day5DSA

    □ DaySDSA
    □ Employee_project
    □ Java_day1
    □ May E System Library [JavaSE-17]
    □ SRE
    □ Set Com.wipro
    □ BuiltinAnno.java
    □ Completriurajava
    □ Control Control Control Control
    □ Completriurajava
    □ Control Control Control
    □ Control Control Control
    □ Control Control Control
    □ Con
                                                                                                                                                                                                 primeNumbers.thenAcceptAsync(result -> {
             customAnnoation.java
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      - X % | R 0 0 0 0 0
                                                                                                                                                               🖺 Problems @ Javadoc 🖳 Declaration 📮 Console ×
                                                                                                                                                              <terminated> Task6 [Java Application] C\Users\vaish\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.10.v20240120-1143\jre\bin\javaw.exe (May 29, 2024, 9:12:11 Finding prime numbers up to 10
             > 🔝 ExecutorExamp.java

    GenLBUB.java
    LinkedListDemo.java

             Manager.java

MyGenClass.java

Salary.java
        > # com.wipro.selectors
>  module-info.java
```

Task 7: Writing Thread-Safe Code, Immutable Objects Design a thread-safe Counter class with increment and decrement methods. Then demonstrate its usage from multiple threads. Also, implement and use an immutable class to share data between threads.

```
package com.assign;
class Counter {
    private int count;
    public synchronized void increment() {
        count++;
    }
    public synchronized void decrement() {
        count--;
    }
    public synchronized int getCount() {
        return count;
    }
}
final class ImmutableData {
    private final String data;
    public ImmutableData(String data) {
        this.data = data;
    }
    public String getData() {
        return data;
    }
}
public class Task7 {
    public static void main(String[] args) {
        Counter counter = new Counter();
        Runnable incrementTask = () -> {
            for (int i = 0; i < 1000; i++) {
                counter.increment();
            }
        };
```

```
Runnable decrementTask = () -> {
            for (int i = 0; i < 1000; i++) {
                counter.decrement();
            }
        };
        Thread thread1 = new Thread(incrementTask);
        Thread thread2 = new Thread(decrementTask);
        thread1.start();
        thread2.start();
        try {
            thread1.join();
            thread2.join();
        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
        }
        System.out.println("Final count: " + counter.getCount());
        // Immutable object usage
        ImmutableData immutableData = new ImmutableData("Some data");
        System.out.println("Immutable data: " +
immutableData.getData());
   }
}
```

```
1 package com.assign;
  class Counter {
private int count;

→ # com.assign

      ## com.assign

> ② BankDemojava

> ② CommunicationDemojava

> ② TaskS.java

> ② TaskS.java

> ② TaskS.java
                                                                      public synchronized void increment() {
                                                                           count++;

ThreadDemo.java

ThreadStateAndTransition.java
                                                             10°
11
                                                                     public synchronized void decrement() {
                                                                          count--;
                                                            11
12
13
14°
15
16
17 }
18
                                                                     }
     > 🗓 module-info.java
> ☑ module-infoJava

> ☑ Day5DSA

> 涵 JRE System Library [JavaSE-17]

> ⑤ src

- ② com.wipro.algo
                                                                     public synchronized int getCount() {
                                                                          return count;
    > ② LinearSearch.java

• ■ com.wipro.dll

> ② Dll.java
    20 final class ImmutableData {
21 private final String data;
       > DigOnsquare.java
                                                            22
    22
23®
24
25
26
                                                                     public ImmutableData(String data) {
    this.data = data;
                                                                    }
    > # com.wipro.q

- # com.wipro.se

- D BinaryDemo.java

    JumpSearch.java
    LinearSearch.java
```