NATIONAL TECHNICAL UNIVERSITY OF UKRAINE
“IGOR SIKORSKY KYIV POLYTECHNIC INSTITUTE”

Faculty of Informatics and Computer Engineering

Department of Computer Engineering

Distributed Information Systems

Lab №5 Inter-thread communication

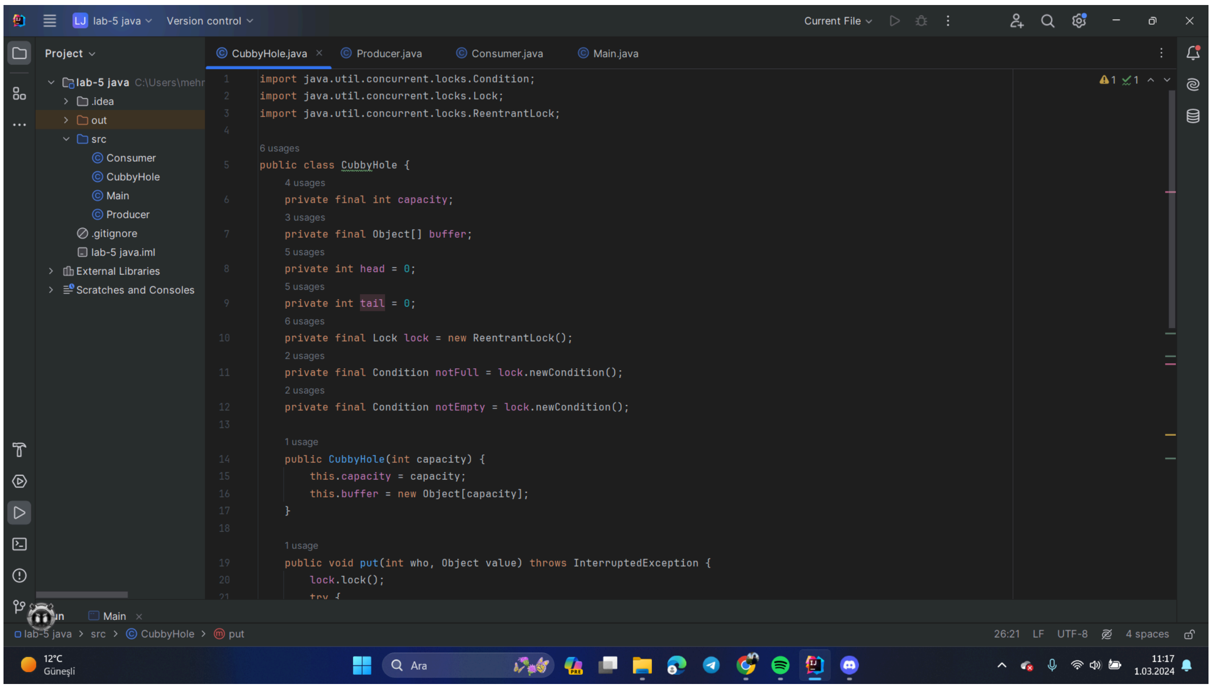
Student, group IM-14 FIOT

MEHMET KULUBECİOGLU

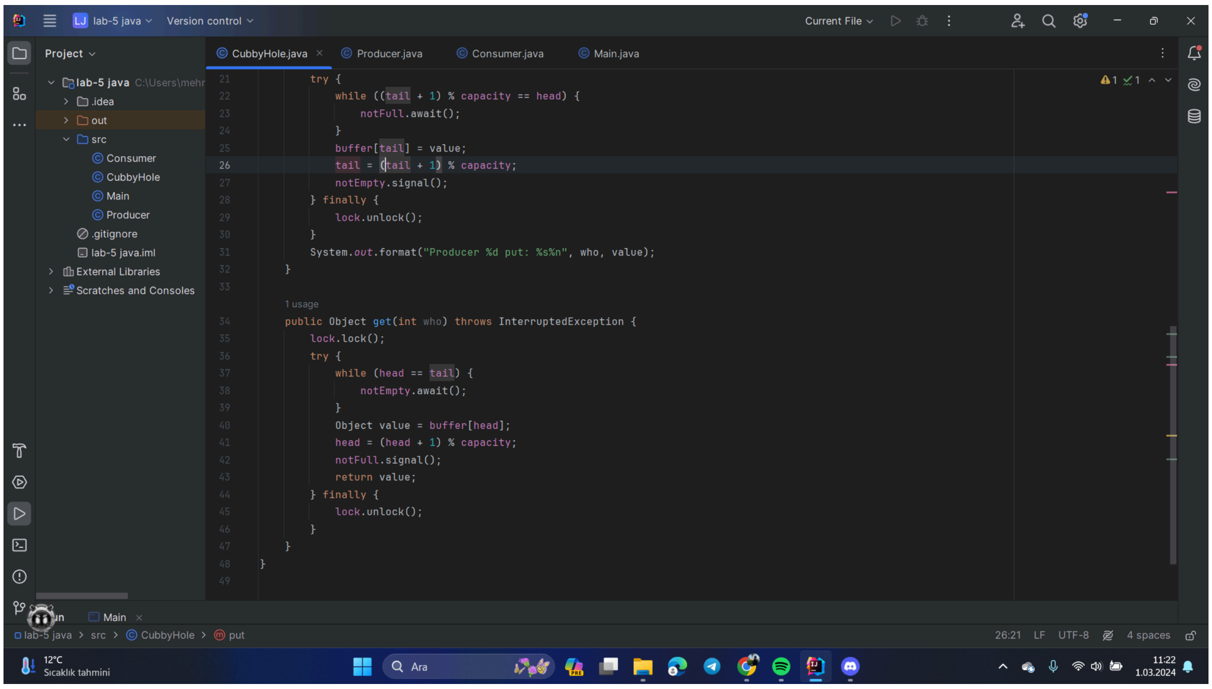
Reviewer YULİA TİMOFEEVA

Kyiv – 2024

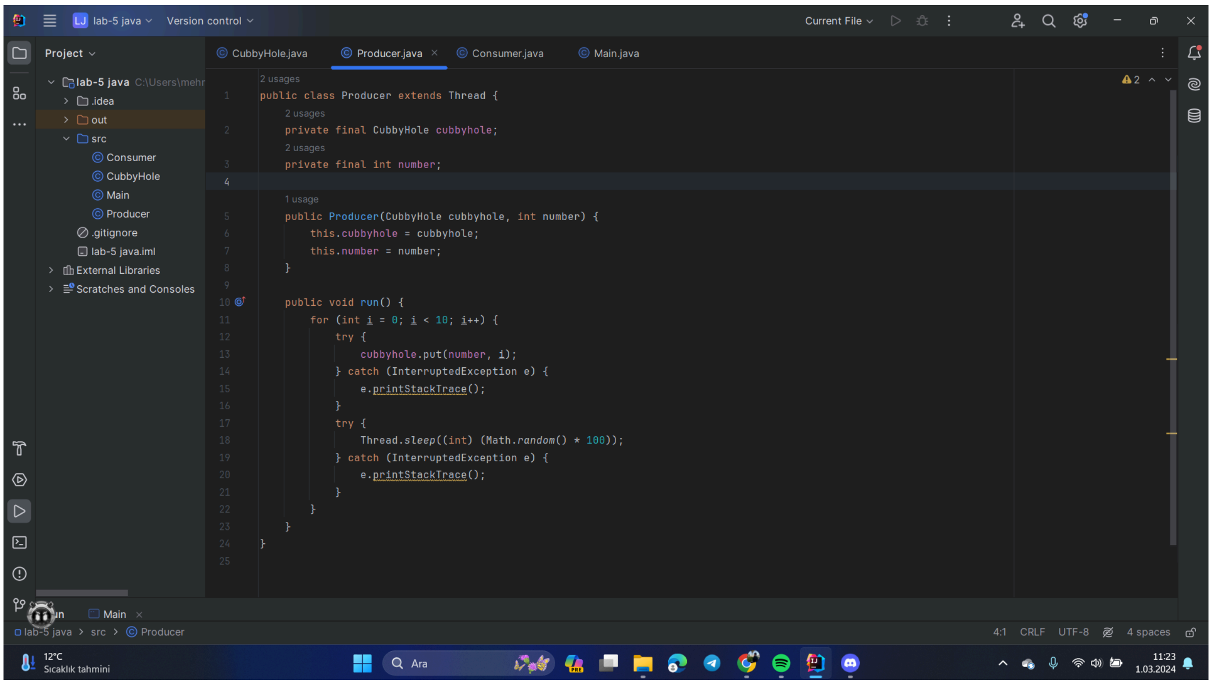
My Full Codes:



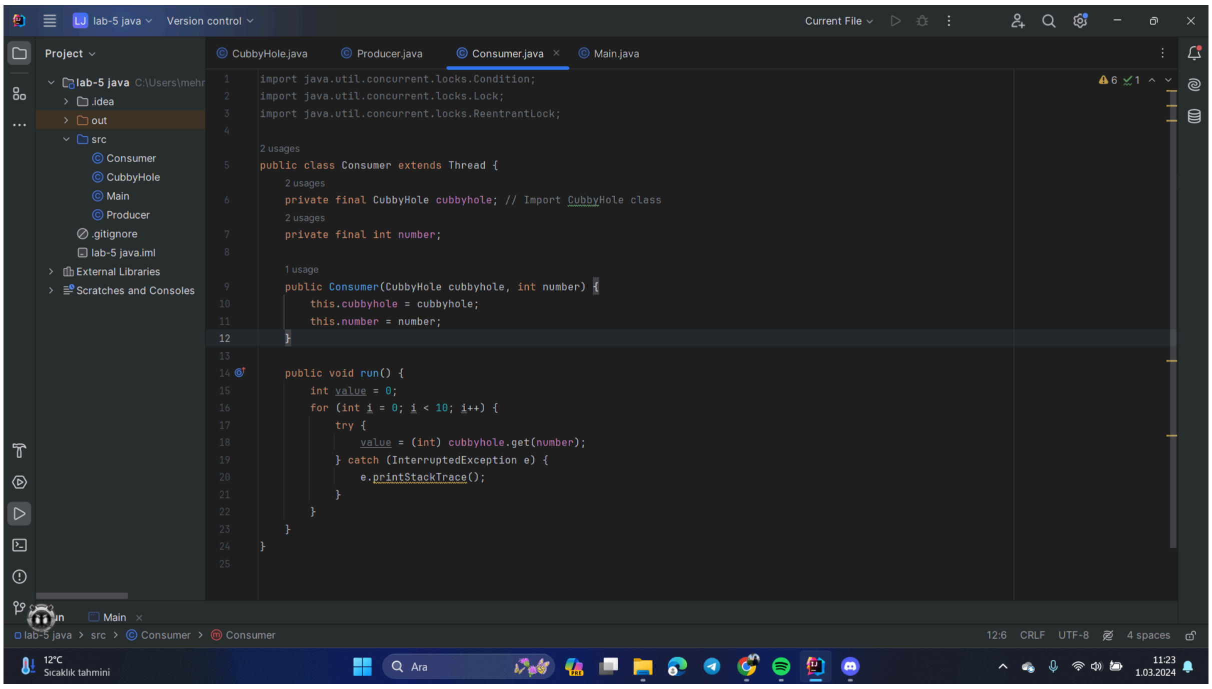
```java
import java.util.concurrent.locks.Condition;
import java.util.concurrent.locks.Lock;
import java.util.concurrent.locks.ReentrantLock;

public class CubbyHole {
    private final int capacity;
    private final Object[] buffer;
    private int head = 0;
    private int tail = 0;
    private final Lock lock = new ReentrantLock();
    private final Condition notFull = lock.newCondition();
    private final Condition notEmpty = lock.newCondition();

    public CubbyHole(int capacity) {
        this.capacity = capacity;
        this.buffer = new Object[capacity];
    }

    public void put(int who, Object value) throws InterruptedException {
        lock.lock();
```



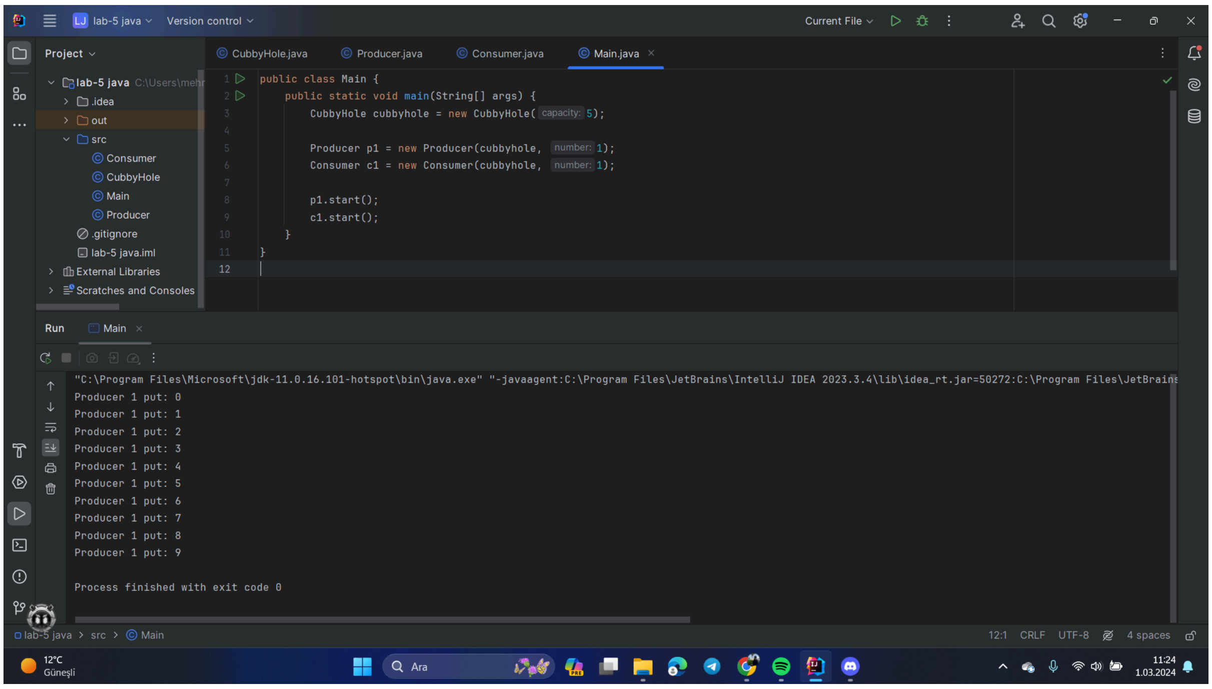
```java
        try {
            while ((tail + 1) % capacity == head) {
                notFull.await();
            }
            buffer[tail] = value;
            tail = (tail + 1) % capacity;
            notEmpty.signal();
        } finally {
            lock.unlock();
        }
        System.out.format("Producer %d put: %s%n", who, value);
    }

    public Object get(int who) throws InterruptedException {
        lock.lock();
        try {
            while (head == tail) {
                notEmpty.await();
            }
            Object value = buffer[head];
            head = (head + 1) % capacity;
            notFull.signal();
            return value;
        } finally {
            lock.unlock();
        }
    }
}
```



```java
2 usages
public class Producer extends Thread {
    2 usages
    private final CubbyHole cubbyhole;
    2 usages
    private final int number;

    1 usage
    public Producer(CubbyHole cubbyhole, int number) {
        this.cubbyhole = cubbyhole;
        this.number = number;
    }

    public void run() {
        for (int i = 0; i < 10; i++) {
            try {
                cubbyhole.put(number, i);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            try {
                Thread.sleep((int) (Math.random() * 100));
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
```



```java
import java.util.concurrent.locks.Condition;
import java.util.concurrent.locks.Lock;
import java.util.concurrent.locks.ReentrantLock;

2 usages
public class Consumer extends Thread {
    2 usages
    private final CubbyHole cubbyhole; // Import CubbyHole class
    2 usages
    private final int number;

    1 usage
    public Consumer(CubbyHole cubbyhole, int number) {
        this.cubbyhole = cubbyhole;
        this.number = number;
    }

    public void run() {
        int value = 0;
        for (int i = 0; i < 10; i++) {
            try {
                value = (int) cubbyhole.get(number);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
```



```java
public class Main {
    public static void main(String[] args) {
        CubbyHole cubbyhole = new CubbyHole(capacity: 5);

        Producer p1 = new Producer(cubbyhole, number: 1);
        Consumer c1 = new Consumer(cubbyhole, number: 1);

        p1.start();
        c1.start();
    }
}
```

```
"C:\Program Files\Microsoft\jdk-11.0.16.101-hotspot\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2023.3.4\lib\idea_rt.jar=50272:C:\Program Files\JetBrains
Producer 1 put: 0
Producer 1 put: 1
Producer 1 put: 2
Producer 1 put: 3
Producer 1 put: 4
Producer 1 put: 5
Producer 1 put: 6
Producer 1 put: 7
Producer 1 put: 8
Producer 1 put: 9

Process finished with exit code 0
```

Producer-Consumer Problem in Java

Purpose:

- These codes address the "Producer-Consumer Problem", which allows multiple threads to use a shared resource in a coordinated manner and transfer data efficiently.

Function of Codes:

The codes consist of the following classes:

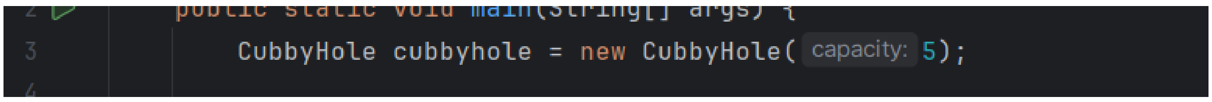
- **CubbyHole:** Represents the shared limited buffer.
- **Producer:** Represents the thread that adds produced values to the buffer.
- **Consumer:** Represents the thread that retrieves values from the buffer.
- **Main:** This class starts the program and creates producer and consumer threads.

Step by Step Description:

1. **CubbyHole Class:**

- The buffer is created: CubbyHole cubbyhole = new CubbyHole(5);

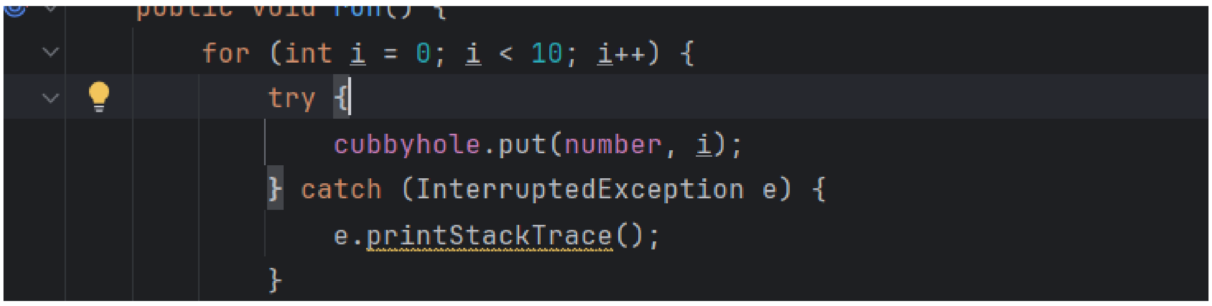
Example:



```
public static void main(String[] args) {
    CubbyHole cubbyhole = new CubbyHole( capacity: 5);
```

- The generator adds the values to the buffer using the put function: cubbyhole.put(1, value);

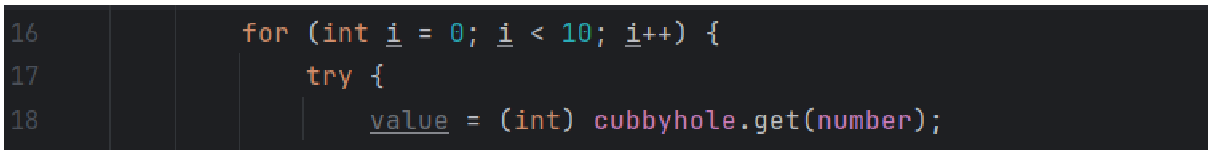
Example:



```
public void run() {
    for (int i = 0; i < 10; i++) {
        try {
            cubbyhole.put(number, i);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
```

- The consumer retrieves values from the buffer using the get function: value = cubbyhole.get(1);

Example:

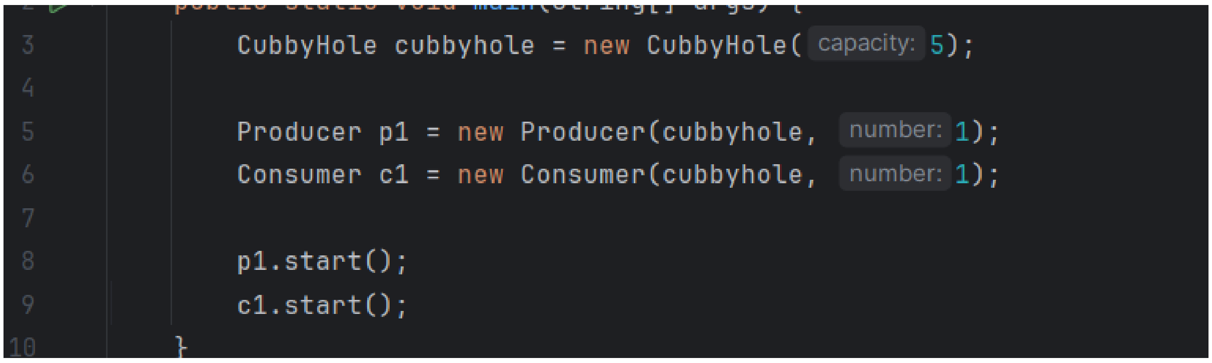


```
    for (int i = 0; i < 10; i++) {
        try {
            value = (int) cubbyhole.get(number);
```

2. **Producer Class:**

- The producer thread is created: Producer producer = new Producer(cubbyhole, 1);

Example:

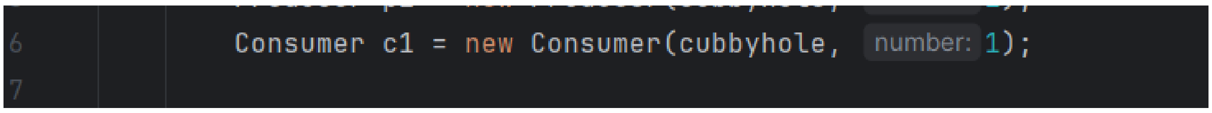


```java
    public static void main(String[] args) {
        CubbyHole cubbyhole = new CubbyHole( capacity: 5);

        Producer p1 = new Producer(cubbyhole,  number: 1);
        Consumer c1 = new Consumer(cubbyhole,  number: 1);

        p1.start();
        c1.start();
    }
```

Consumer Class:

- The consumer thread is created: Consumer c1 = new Consumer(cubbyhole, 1);

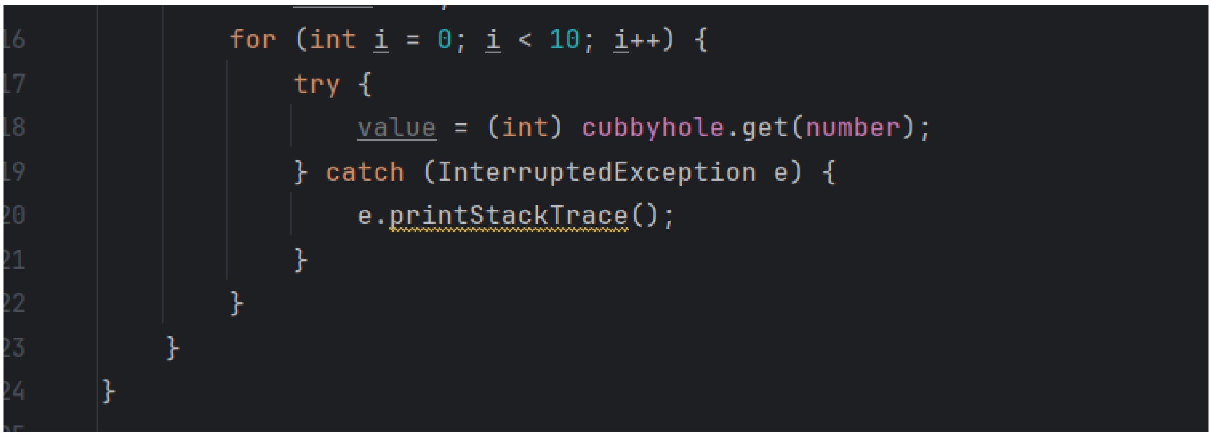
Example:



```java
        Consumer c1 = new Consumer(cubbyhole,  number: 1);
```

- Consumer retrieves values from buffer 10 times: consumer.run();

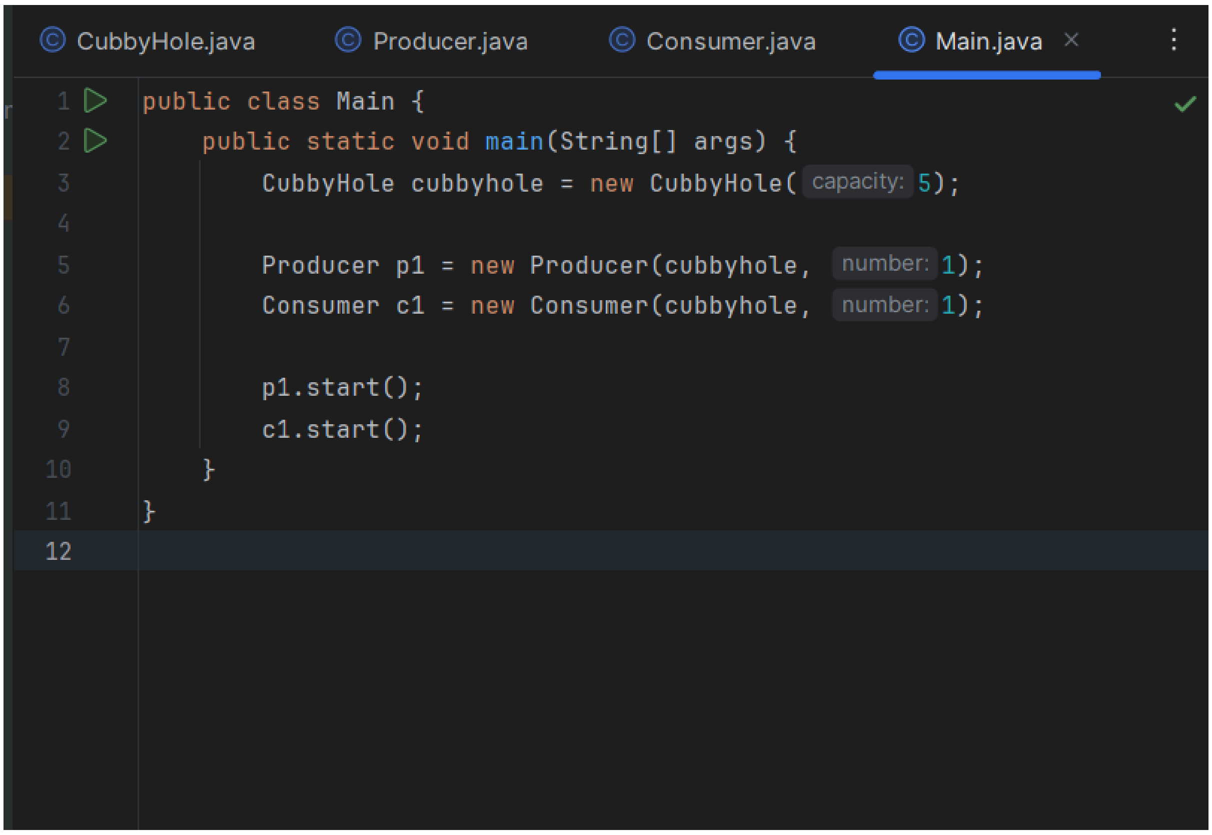
Example:



```java
            for (int i = 0; i < 10; i++) {
                try {
                    value = (int) cubbyhole.get(number);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        }
    }
```

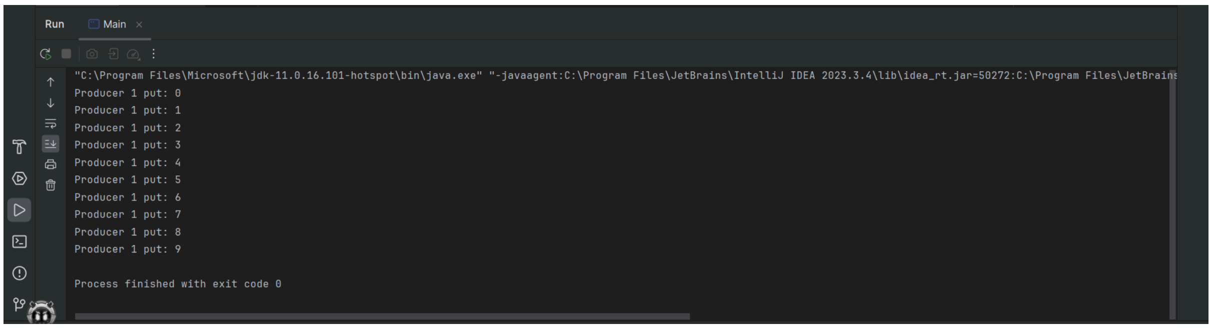
Main Class:

- Buffer and producer-consumer threads are created:



```java
public class Main {
    public static void main(String[] args) {
        CubbyHole cubbyhole = new CubbyHole( capacity: 5);

        Producer p1 = new Producer(cubbyhole,  number: 1);
        Consumer c1 = new Consumer(cubbyhole,  number: 1);

        p1.start();
        c1.start();
    }
}
```

Output:



```
"C:\Program Files\Microsoft\jdk-11.0.16.101-hotspot\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2023.3.4\lib\idea_rt.jar=50272:C:\Program Files\JetBrains
Producer 1 put: 0
Producer 1 put: 1
Producer 1 put: 2
Producer 1 put: 3
Producer 1 put: 4
Producer 1 put: 5
Producer 1 put: 6
Producer 1 put: 7
Producer 1 put: 8
Producer 1 put: 9

Process finished with exit code 0
```