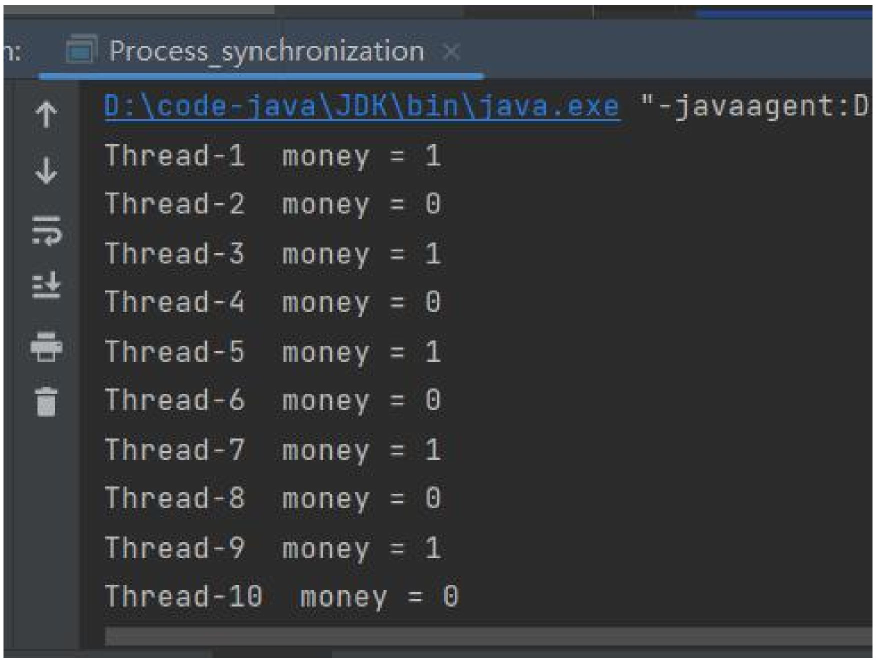
Flow

1. Create class ‘Mythread’ which extends Thread. Each class has id which distinguishes odd thread from even thread. Each class couldn’t operate until the previous thread finishes
2. Create 10 ‘Mythread’ objects and appoint previous thread for every thread expect the first one and operate ‘run’ function.
3. The ‘run’ function first operates ‘P’ function. In ‘P’ function, the ‘join’ function could let one thread wait for another thread.
4. The first thread directly operates and increases the money. Then the later thread could operate one by one. The order is “1,2,3⋯⋯10”. The money is either 1 or 0.

Code

```java
public class Process_synchronization {
    static int money=0;

    public static void main(String[] args) {
        Mythread t,pre=null;
        for(int i=1;i<=10;i++)
        {
            t=new Mythread(i);
            t.set(pre); //set previous thread for every thread
            t.start();
            pre=t;
        }
    }
}

class Mythread extends Thread{
    int id;
    Thread pre;
    int d=1;
    public Mythread(int id)
    {
        this.id=id;
        if(id%2==0)
            d=-1;
    }

    public void set(Thread t) {
        pre=t;
    }
    @Override
    public void run() {
        P();
        V();
    }

    public void P()
    {
        try{
            if(pre!=null)
                pre.join();
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
    }

    public void V()
    {
        Process_synchronization.money+=d;
        System.out.println("Thread-" + id+ "    money = " + Process_synchronization.money);
    }
}
```

The result of running



```
D:\code-java\JDK\bin\java.exe "-javaagent:D
Thread-1  money = 1
Thread-2  money = 0
Thread-3  money = 1
Thread-4  money = 0
Thread-5  money = 1
Thread-6  money = 0
Thread-7  money = 1
Thread-8  money = 0
Thread-9  money = 1
Thread-10  money = 0
```