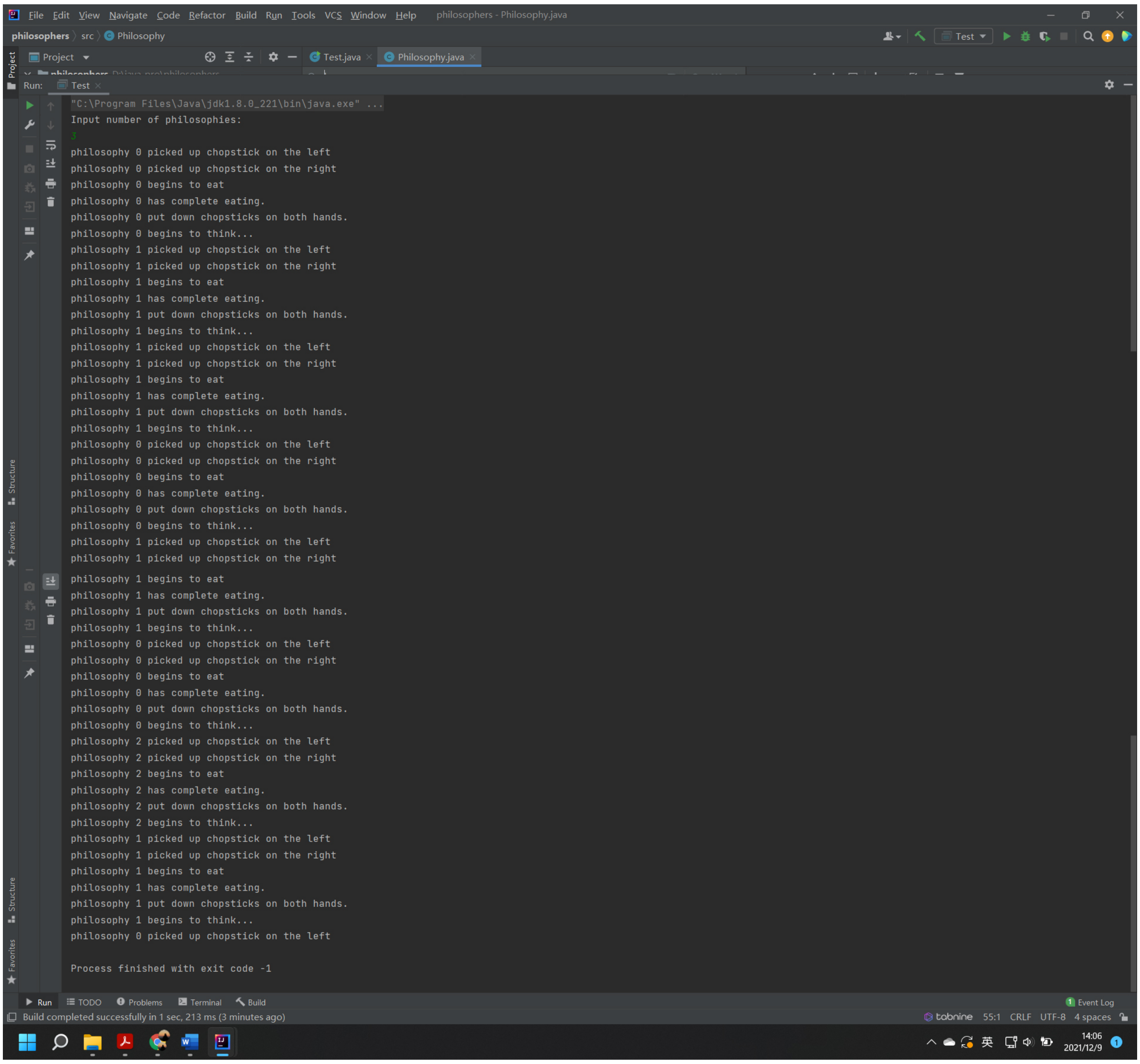
Sen Wang_19722109

N philosophers sit around and eating model solution:

First, i will attach the console result.



File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
philosophers - Philosophy.java
philosophers 〉 src 〉 Philosophy
Project
Test.java
Philosophy.java
Run:
Test
Run
TODO
Problems
Terminal
Build
Event Log
tabnine 55:1 CRLF UTF-8 4 spaces
14:06
2021/12/9

```
"C:\Program Files\Java\jdk1.8.0_221\bin\java.exe" ...
Input number of philosophies:
3
philosophy 0 picked up chopstick on the left
philosophy 0 picked up chopstick on the right
philosophy 0 begins to eat
philosophy 0 has complete eating.
philosophy 0 put down chopsticks on both hands.
philosophy 0 begins to think...
philosophy 1 picked up chopstick on the left
philosophy 1 picked up chopstick on the right
philosophy 1 begins to eat
philosophy 1 has complete eating.
philosophy 1 put down chopsticks on both hands.
philosophy 1 begins to think...
philosophy 1 picked up chopstick on the left
philosophy 1 picked up chopstick on the right
philosophy 1 begins to eat
philosophy 1 has complete eating.
philosophy 1 put down chopsticks on both hands.
philosophy 1 begins to think...
philosophy 0 picked up chopstick on the left
philosophy 0 picked up chopstick on the right
philosophy 0 begins to eat
philosophy 0 has complete eating.
philosophy 0 put down chopsticks on both hands.
philosophy 0 begins to think...
philosophy 1 picked up chopstick on the left
philosophy 1 picked up chopstick on the right
philosophy 1 begins to eat
philosophy 1 has complete eating.
philosophy 1 put down chopsticks on both hands.
philosophy 1 begins to think...
philosophy 0 picked up chopstick on the left
philosophy 0 picked up chopstick on the right
philosophy 0 begins to eat
philosophy 0 has complete eating.
philosophy 0 put down chopsticks on both hands.
philosophy 0 begins to think...
philosophy 2 picked up chopstick on the left
philosophy 2 picked up chopstick on the right
philosophy 2 begins to eat
philosophy 2 has complete eating.
philosophy 2 put down chopsticks on both hands.
philosophy 2 begins to think...
philosophy 1 picked up chopstick on the left
philosophy 1 picked up chopstick on the right
philosophy 1 begins to eat
philosophy 1 has complete eating.
philosophy 1 put down chopsticks on both hands.
philosophy 1 begins to think...
philosophy 0 picked up chopstick on the left

Process finished with exit code -1
```

Build completed successfully in 1 sec, 213 ms (3 minutes ago)

As you can see, you can set the number of philosophers at the beginning. All the philosophers' eating and thinking can continue until the user chooses to close the running program.

However, the small problem with this is that sometimes there is process hunger, e.g. Philosopher #2 starts eating late. If we add a variable A, we can set the eaten philosopher's "hunger" value to "false"(). This variable is reflected in the comments.

Next, I will briefly introduce my philosopher Class.

The main method is that before check chopstick states, get lock, then if both left and right chopsticks are free, phisolopher can begin to eat. After eating, free both chopsticks.

```
public class Philosophy extends Thread{
    boolean [] chopsticks; //true:taken      false: untaken
    private int i;
    private Boolean leftFlag=false;  //whether the philosophy has picked up the chopstick
    private Boolean rightFlag=false;
    private Boolean hungry=true; //true： hasn't taken food    false： has taken
food
    private Boolean thinkFlag=false;  //false： not thinking  true： thinking
    private Object obj;
    public Philosophy(int i , Boolean leftFlag , Boolean rightFlag , Boolean
hungry ,Boolean thinkFlag)
    {
        this.i=i;
        this.leftFlag=leftFlag;
        this.rightFlag=rightFlag;
        this.hungry = hungry;
        this.thinkFlag=thinkFlag;
    }
    public void run(){
        while(true){
            this.eat();
            this.think();
        }
    }
    public void eat(){
        synchronized (obj)
        {
            //left and right chopsticks are both untaken,

 if(!chopsticks[this.i]&&!chopsticks[(this.i+1)%Test.a]&&!this.leftFlag&&!this.r
ightFlag)//&&this.hungry
                //this philosophy can pick up left chopstick and right chopstick
            {
                try {
                    Thread.sleep(1000);
                } catch (InterruptedException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
                chopsticks[(this.i+1)%Test.a]=true;    //chopstick on right side
be picked up
                this.leftFlag=true;     //left hand with chopstick
                System.out.println("philosophy "+this.i+" picked up chopstick on
the left");
                try {
                    Thread.sleep(1000);
                } catch (InterruptedException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
                chopsticks[this.i]=true;
                this.rightFlag=true;
                System.out.println("philosophy "+this.i+" picked up chopstick on
the right");
            }
        }
        if(this.leftFlag&&this.rightFlag)//&&this.hungry
        {
            System.out.println("philosophy "+this.i+" begins to eat");
            this.thinkFlag=false; //stop thinking
            try {
                Thread.sleep(20000);
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
            System.out.println("philosophy "+this.i+" has complete eating.");
            this.hungry =false;
            try {
                Thread.sleep(2000);
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
            System.out.println("philosophy "+this.i+" put down chopsticks on both hands.");
            System.out.println("philosophy "+this.i+" begins to think...");
            this.thinkFlag=true;  //begin to think
            this.leftFlag=false;
            this.rightFlag=false;
            chopsticks[i]=false;
            chopsticks[(i+1)%Test.a]=false;
        }
    }
    public void think(){
        if(!this.leftFlag&&!this.rightFlag&&!this.thinkFlag)
        {
            thinkFlag=true;
            System.out.println("philosophy "+i+" begins to think...");
        }
    }
    public void setobj(Object obj) {
        // TODO Auto-generated method stub
        this.obj=obj;
    }
    public void setchopsticks(boolean[] chopsticks) {
        // TODO Auto-generated method stub
        this.chopsticks=chopsticks;
    }
}
```