Qian Wang

COP 4600

1089

10/15/2015

HW7

On my honor, I have neither given nor received unauthorized aid in doing this assignment.

Qian Wang

Summary:

The easiest parts?

Given taking Design Pattern at the moment, I quickly realized State Pattern is great for this homework.

The most difficult parts?

Trying to make sure if my code is working is the most difficult part.

The assignment's educational objectives?

Understanding State Pattern and more real life example practice with Thread and its library

How well did I achieved them?

I believe I did the homework's requirements.

Qian Wang

COP 4600

1089

10/15/2015

HW7

1. Does the program compile without errors?

   No. This code has no error.

2. Does the program compile without warnings?

   No. This code has no warnings.

3. Does the program run without crashing?

   No. This code run without crashing.

4. Describe how you tested the program.

   I had tried serval test case to make sure that every state is included in my code. And outputting detail message to let me know where is my mistake.

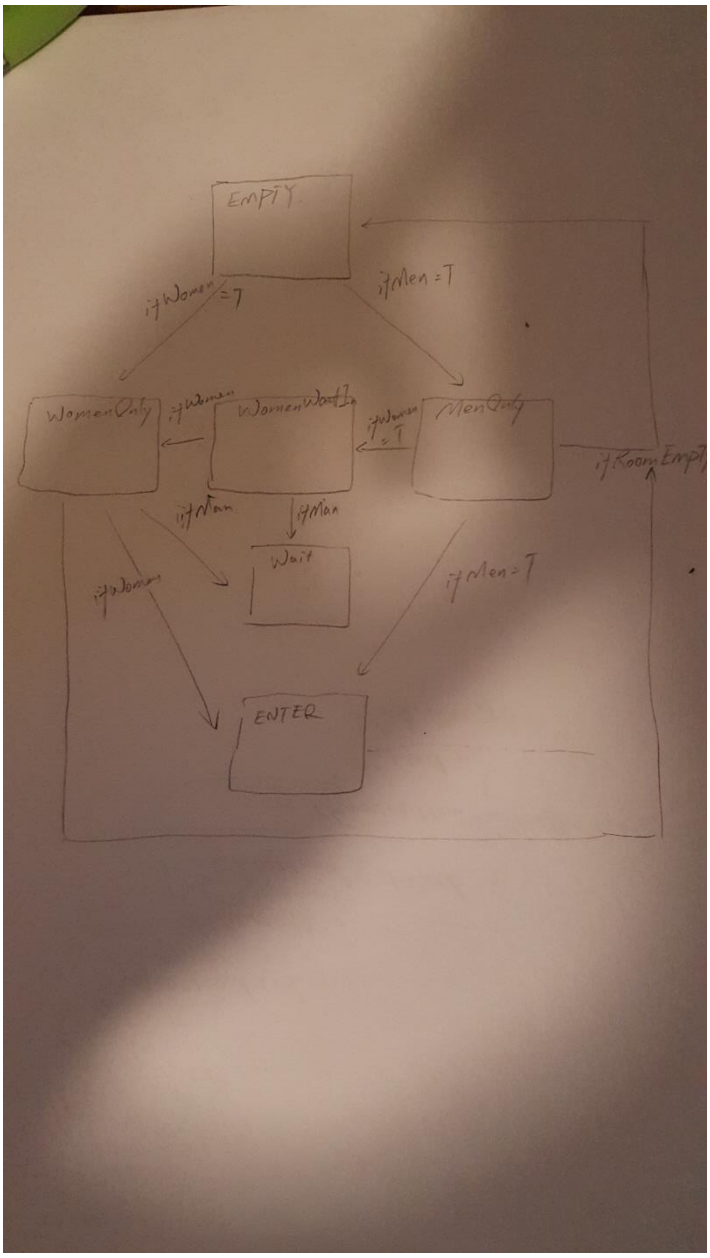
5. Describe the ways in which the program does *not* meet assignment's specifications.

   I am say the only part that my program doesn't meet the specifications is that I have one extra statement than the book is said.

6. Does the program run correctly?

   Yes

State:



Run through:

When first create, the bathroom is at empty state. Depends on the next human's sex, state will become MenOnly or WomenOnly.

When in MenOnly, any man can enter. But a women will change the state to WomenWantIn

When in WomenWantIn state, and Man enter will be waiting, after the room's empty it become womenOnly state.

When in womenOnly state, Man simply wait and Women will enter.

During MenOnly State and WomenOnly state, when the room is empty it become empty state again.

Last 100 output:

Woman 7 wants to enter

Man 10 is done do bussinessing

Man is trying to leave

Man 10 left

Man 10 is begining to recharge liquid for 4774 milliseconds

Woman 8 is done recharge liquiding

Woman 8 wants to enter

Man 11 is done do bussinessing

Man is trying to leave

Man 11 left

Man 11 is begining to recharge liquid for 1686 milliseconds

Man 17 is done recharge liquiding

Man 17 wants to enter

Be a gentleman, let the lady go first

Man 19 is done do bussinessing

Man is trying to leave

Man 19 left

Man 19 is begining to recharge liquid for 3446 milliseconds

Be a gentleman, let the lady go first

Woman 1 is done recharge liquiding

Woman 1 wants to enter

Man 14 is done do bussinessing

Man is trying to leave

Man 14 left

Man 14 is begining to recharge liquid for 3251 milliseconds

Be a gentleman, let the lady go first

Man 12 is done do bussinessing

Man is trying to leave

Man 12 left

Man 12 is begining to recharge liquid for 1787 milliseconds

Be a gentleman, let the lady go first

Man 18 is done do bussinessing

Man is trying to leave

Man 18 left

Man 18 is begining to recharge liquid for 4417 milliseconds

Be a gentleman, let the lady go first

Man 13 is done do bussinessing

Man is trying to leave

Man 13 left

Man 13 is begining to recharge liquid for 3145 milliseconds

Be a gentleman, let the lady go first

Woman 7 is in

Analysis:

The above output shows that, when woman 7 wants to enter before man 10 is done. That action cause all the rest of the man who wants to enter before woman 7 to be a gentleman and let Woman 7 enter first.

Output:

Woman 3Spend 22398 inside Bathroom and 29631 outBathroom recharging herself with liquid

Woman 2Spend 27212 inside Bathroom and 29988 outBathroom recharging herself with liquid

Woman 6Spend 27946 inside Bathroom and 27610 outBathroom recharging herself with liquid

Woman 8Spend 31622 inside Bathroom and 26404 outBathroom recharging herself with liquid

Woman 7Spend 29479 inside Bathroom and 27148 outBathroom recharging herself with liquid

Man 17Spend 1171 inside Bathroom and 2405 outBathroom recharging himself with liquid

Man 19Spend 3795 inside Bathroom and 3446 outBathroom recharging himself with liquid

Man 13Spend 4735 inside Bathroom and 3145 outBathroom recharging himself with liquid

Woman 0Spend 28843 inside Bathroom and 26075 outBathroom recharging herself with liquid

Man 18Spend 4237 inside Bathroom and 4417 outBathroom recharging himself with liquid

Woman 4Spend 31858 inside Bathroom and 24039 outBathroom recharging herself with liquid

Woman 9Spend 26375 inside Bathroom and 26194 outBathroom recharging herself with liquid

Woman 1Spend 32407 inside Bathroom and 25678 outBathroom recharging herself with liquid

Man 15Spend 1974 inside Bathroom and 4026 outBathroom recharging himself with liquid

Woman 5Spend 20758 inside Bathroom and 36454 outBathroom recharging herself with liquid

Man 10Spend 2498 inside Bathroom and 4774 outBathroom recharging himself with liquid

Man 12Spend 4165 inside Bathroom and 1787 outBathroom recharging himself with liquid

Man 14Spend 4113 inside Bathroom and 3251 outBathroom recharging himself with liquid

Man 11Spend 3338 inside Bathroom and 1686 outBathroom recharging himself with liquid

Man 16Spend 1053 inside Bathroom and 4448 outBathroom recharging himself with liquid

Analysis: We can clearly see that woman spend more time inside of the bathroom than a man.

Code:

```
import java.util.Random;

class UnisexRestroom{
        private State currentState;
        private Human[] human;
        private int menIn;
        private int womenIn;
        private Object lock = new Object();
   static private Random random = new Random();

        public static void main(String[] arg){
                new UnisexRestroom(10, 10);
        }

        public UnisexRestroom(int man, int woman){
                initialize(man, woman);
                startSimulation();
                sleep(60000);
                shutdownSimulation();
        }
```

```
	private void sleep( int duration ){
	   try
	   {
	     Thread.currentThread().sleep( duration );
	   }
	   catch ( InterruptedException e )
	   {
	     Thread.currentThread().interrupt();
	   }
	}
	private void shutdownSimulation(){
		for(int i = 0; i != human.length; ++i){
			human[i].interrupt();
		}
	}
	private void initialize(int man, int woman){
		//initialize a array of users who is going to use the bathroom;
		this.human = new Human[man + woman];
		for(int i = 0; i != man + woman; ++i){
			for(int k = 0; k != woman; ++k){
				this.human[k] = new Woman(k);
			}
			for(int j = woman; j != man + woman; ++j){
				this.human[j] = new Man(j);
			}
		}
		this.currentState = new Empty();
	}

	private void startSimulation(){
		int n = human.length;

		System.out.println("There are " + n + " human planing using the bathrom and " +
menIn + " men in the bathroom");
		for(int i = 0; i != n; ++i){
			human[i].start();
		}
	}

	public State getState(){
		return currentState;
	}

	class Woman extends Human{
```

```
private String name;
private int inBathroom = 0;
private int outBathroom = 0;

public Woman(int i){
	this.name = "Woman " + i;
}
public void wantsEnter() throws InterruptedException{
	synchronized(lock){
		System.out.println(name + " wants to enter");
		while(!currentState.wantsEnter()){
			lock.wait();
		}

		womenIn ++;
		System.out.println(name + " is in");
	}
}
public void leave(){
	synchronized(lock){
		currentState.leave();
		System.out.println(name + " left");

		lock.notifyAll();
	}
}


public void run(){
	try{
		while(!interrupted()){
			wantsEnter();
			inBathroom += doAction("do bussiness");
			leave();
			//I need time to charge up my bladder
			outBathroom += doAction("recharge liquid");

		}

	}
	catch(InterruptedException e){
		System.out.println(printResult());
	}
}
```

```
        private int doAction( String act ) throws InterruptedException{
           int time = random.nextInt( 4000 ) + 1000 ;
           System.out.println( name + " is begining to " + act + " for " + time +
                                   " milliseconds" );
           sleep( time );

           System.out.println( name + " is done " + act + "ing" );

           return time;
        }
        public String toString(){
                return "Woman";
        }
        public String printResult(){
                return name + "Spend " + inBathroom + " inside Bathroom and " +
outBathroom + " outBathroom recharging herself with liquid";
        }
    }

    class Man extends Human{
        private String name;
        private int inBathroom = 0;
        private int outBathroom = 0;

        public Man(int i){
                this.name = "Man " + i;
        }
        public void wantsEnter() throws InterruptedException{
                synchronized(lock){
                        System.out.println(name + " wants to enter");
                        while(!currentState.wantsEnter()){
                                lock.wait();
                        }

                        menIn ++;
                        System.out.println(name + " is in");
                }

        }
        private int doAction( String act ) throws InterruptedException{
           int time = random.nextInt( 4000 ) + 1000 ;
           System.out.println( name + " is begining to " + act + " for " + time +
                                   " milliseconds" );
           sleep( time );
```

```
            System.out.println( name + " is done " + act + "ing" );

            return time;
        }

        public void leave(){
            synchronized(lock){
                currentState.leave();
                System.out.println(name + " left");

                lock.notifyAll();
            }
        }

        public void run(){
            try{
                while(!interrupted()){
                    wantsEnter();
                    inBathroom += doAction("do bussiness");
                    leave();
                    outBathroom += doAction("recharge liquid");
                }

            }
            catch(InterruptedException e){
                System.out.println(printResult());
            }
        }

        public String toString(){
            return "Man";
        }
        public String printResult(){
            return name + "Spend " + inBathroom + " inside Bathroom and " +
outBathroom + " outBathroom recharging himself with liquid";
        }
    }


    class Empty extends State{
        public boolean wantsEnter(){
            synchronized(lock){
                if(((Human)Thread.currentThread()).toString() == "Woman"){
```

```
                                        currentState = new WomenOnly();
                                        return true;

                                }
                                else if(((Human)Thread.currentThread()).toString() == "Man"){
                                        currentState = new ManOnly();
                                        return true;
                                }
                                else{
                                        throw new RuntimeException("Some other gender tries to
enter");
                                }
                        }

                }
                public String toString(){
                        return "Empty";
                }
                public void leave(){
                        synchronized(lock){
                                System.out.println("There is no one here!");
                                lock.notifyAll();
                        }

                }

        }
        class ManOnly extends State{
                public boolean wantsEnter(){
                        synchronized(lock){
                                if(womenIn == 0 && menIn == 0){
                                        currentState = new Empty();
                                        return true;
                                }
                                else if(((Human)Thread.currentThread()).toString() == "Man"){
                                        return true;
                                }
                                else if(((Human)Thread.currentThread()).toString() == "Woman"){
                                        currentState = new WomenWantIn();
                                        return false;
                                }
                                else{
                                        throw new RuntimeException("Some other gender tries to
enter");
```

```
                        }
                    }

                }
                public String toString(){
                    return "ManOnly";
                }
                public void leave(){
                    synchronized(lock){

                        if(((Human)Thread.currentThread()).toString() == "Man" &&
menIn != 0){
                            --menIn;
                        }
                        else{
                            throw new RuntimeException("why are you here?");
                        }
                    }

                }
            }
            class WomenWantIn extends State{
                public boolean wantsEnter(){
                    synchronized(lock){
                        if(((Human)Thread.currentThread()).toString() == "Man"){
                            System.out.println("Be a gentleman, let the lady go first");
                            return false;
                        }
                        else if(((Human)Thread.currentThread()).toString() == "Woman"){
                            if(womenIn == 0 && menIn == 0){
                                currentState = new WomenOnly();
                                return true;
                            }
                            else{
                                return false;
                            }
                        }
                        else{
                            throw new RuntimeException("Some other gender tries to
enter");
                        }
                    }

                }
```

```
		public String toString(){
			return "WomenWantIn";
		}

		public void leave(){
			synchronized(lock){
				System.out.println((Human)Thread.currentThread() + " is trying to
leave");
				if(womenIn == 0 && menIn == 0){
					currentState = new WomenOnly();
				}
				else if(((Human)Thread.currentThread()).toString() == "Man" &&
menIn != 0){
					--menIn;
				}
				else{
					throw new RuntimeException("why are you here?");
				}
			}

		}
	}
	class WomenOnly extends State{
		public boolean wantsEnter(){
			synchronized(lock){
				if(womenIn == 0 && menIn == 0){
					currentState = new Empty();
					return true;
				}
				else if(((Human)Thread.currentThread()).toString() == "Woman"){
					return true;

				}
				else if(((Human)Thread.currentThread()).toString() == "Man"){
					return false;
				}
				else{
					throw new RuntimeException("Some other gender tries to
enter");
				}
			}

		}
```

```
                public String toString(){
                        return "WomenOnly";
                }

                public void leave(){
                        synchronized(lock){
                                System.out.println((Human)Thread.currentThread() + " is trying to
leave");
                                if(womenIn == 0 && menIn == 0){
                                        currentState = new Empty();
                                }
                                else if(((Human)Thread.currentThread()).toString() == "Woman"
&& womenIn != 0){
                                        --womenIn;
                                }
                                else{
                                        throw new RuntimeException("why are you here?");
                                }
                        }
                }
        }


//      define types
        abstract class Human extends Thread{
                abstract protected void wantsEnter()  throws InterruptedException;
                abstract protected void leave() throws InterruptedException;
                public String toString(){
                        return "";
                };
                abstract protected String printResult();
        }
        abstract class State{
                abstract public boolean wantsEnter();
                abstract public void leave();
                public String toString(){
                        return "";
                }
        }
}
```