import java.util.LinkedList;

/\*\*

\* 仓库类Storage实现缓冲区

\*

\* Email:530025983@qq.com

\*

\* @author MONKEY.D.MENG 2011-03-15

\*

\*/

public class Storage

{

// 仓库最大存储量

private final int MAX\_SIZE = 100;

// 仓库存储的载体

private LinkedList<Object> list = new LinkedList<Object>();

// 生产num个产品

public void produce(int num)

{

// 同步代码段

synchronized (list)

{

// 如果仓库剩余容量不足

while (list.size() + num > MAX\_SIZE)

{

System.out.println("【要生产的产品数量】:" + num + "/t【库存量】:"

+ list.size() + "/t暂时不能执行生产任务!");

try

{

// 由于条件不满足，生产阻塞

list.wait();

}

catch (InterruptedException e)

{

e.printStackTrace();

}

}

// 生产条件满足情况下，生产num个产品

for (int i = 1; i <= num; ++i)

{

list.add(new Object());

}

System.out.println("【已经生产产品数】:" + num + "/t【现仓储量为】:" + list.size());

list.notifyAll();

}

}

// 消费num个产品

public void consume(int num)

{

// 同步代码段

synchronized (list)

{

// 如果仓库存储量不足

while (list.size() < num)

{

System.out.println("【要消费的产品数量】:" + num + "/t【库存量】:"

+ list.size() + "/t暂时不能执行生产任务!");

try

{

// 由于条件不满足，消费阻塞

list.wait();

}

catch (InterruptedException e)

{

e.printStackTrace();

}

}

// 消费条件满足情况下，消费num个产品

for (int i = 1; i <= num; ++i)

{

list.remove();

}

System.out.println("【已经消费产品数】:" + num + "/t【现仓储量为】:" + list.size());

list.notifyAll();

}

}

// get/set方法

public LinkedList<Object> getList()

{

return list;

}

public void setList(LinkedList<Object> list)

{

this.list = list;

}

public int getMAX\_SIZE()

{

return MAX\_SIZE;

}

}

/\*\*

\* 生产者类Producer继承线程类Thread

\*

\* Email:530025983@qq.com

\*

\* @author MONKEY.D.MENG 2011-03-15

\*

\*/

public class Producer extends Thread

{

// 每次生产的产品数量

private int num;

// 所在放置的仓库

private Storage storage;

// 构造函数，设置仓库

public Producer(Storage storage)

{

this.storage = storage;

}

// 线程run函数

public void run()

{

produce(num);

}

// 调用仓库Storage的生产函数

public void produce(int num)

{

storage.produce(num);

}

// get/set方法

public int getNum()

{

return num;

}

public void setNum(int num)

{

this.num = num;

}

public Storage getStorage()

{

return storage;

}

public void setStorage(Storage storage)

{

this.storage = storage;

}

}

/\*\*

\* 消费者类Consumer继承线程类Thread

\*

\* Email:530025983@qq.com

\*

\* @author MONKEY.D.MENG 2011-03-15

\*

\*/

public class Consumer extends Thread

{

// 每次消费的产品数量

private int num;

// 所在放置的仓库

private Storage storage;

// 构造函数，设置仓库

public Consumer(Storage storage)

{

this.storage = storage;

}

// 线程run函数

public void run()

{

consume(num);

}

// 调用仓库Storage的生产函数

public void consume(int num)

{

storage.consume(num);

}

// get/set方法

public int getNum()

{

return num;

}

public void setNum(int num)

{

this.num = num;

}

public Storage getStorage()

{

return storage;

}

public void setStorage(Storage storage)

{

this.storage = storage;

}

}

/\*\*

\* 测试类Test

\*

\* Email:530025983@qq.com

\*

\* @author MONKEY.D.MENG 2011-03-15

\*

\*/

public class Test

{

public static void main(String[] args)

{

// 仓库对象

Storage storage = new Storage();

// 生产者对象

Producer p1 = new Producer(storage);

Producer p2 = new Producer(storage);

Producer p3 = new Producer(storage);

Producer p4 = new Producer(storage);

Producer p5 = new Producer(storage);

Producer p6 = new Producer(storage);

Producer p7 = new Producer(storage);

// 消费者对象

Consumer c1 = new Consumer(storage);

Consumer c2 = new Consumer(storage);

Consumer c3 = new Consumer(storage);

// 设置生产者产品生产数量

p1.setNum(10);

p2.setNum(10);

p3.setNum(10);

p4.setNum(10);

p5.setNum(10);

p6.setNum(10);

p7.setNum(80);

// 设置消费者产品消费数量

c1.setNum(50);

c2.setNum(20);

c3.setNum(30);

// 线程开始执行

c1.start();

c2.start();

c3.start();

p1.start();

p2.start();

p3.start();

p4.start();

p5.start();

p6.start();

p7.start();

}

}

【要消费的产品数量】:50 【库存量】:0 暂时不能执行生产任务!

【要消费的产品数量】:30 【库存量】:0 暂时不能执行生产任务!

【要消费的产品数量】:20 【库存量】:0 暂时不能执行生产任务!

【已经生产产品数】:10 【现仓储量为】:10

【要消费的产品数量】:20 【库存量】:10 暂时不能执行生产任务!

【要消费的产品数量】:30 【库存量】:10 暂时不能执行生产任务!

【要消费的产品数量】:50 【库存量】:10 暂时不能执行生产任务!

【已经生产产品数】:10 【现仓储量为】:20

【要消费的产品数量】:50 【库存量】:20 暂时不能执行生产任务!

【要消费的产品数量】:30 【库存量】:20 暂时不能执行生产任务!

【已经消费产品数】:20 【现仓储量为】:0

【已经生产产品数】:10 【现仓储量为】:10

【已经生产产品数】:10 【现仓储量为】:20

【已经生产产品数】:80 【现仓储量为】:100

【要生产的产品数量】:10 【库存量】:100 暂时不能执行生产任务!

【已经消费产品数】:30 【现仓储量为】:70

【已经消费产品数】:50 【现仓储量为】:20

【已经生产产品数】:10 【现仓储量为】:30

【已经生产产品数】:10 【现仓储量为】:40

import java.util.LinkedList;

import java.util.concurrent.locks.Condition;

import java.util.concurrent.locks.Lock;

import java.util.concurrent.locks.ReentrantLock;

/\*\*

\* 仓库类Storage实现缓冲区

\*

\* Email:530025983@qq.com

\*

\* @author MONKEY.D.MENG 2011-03-15

\*

\*/

public class Storage

{

// 仓库最大存储量

private final int MAX\_SIZE = 100;

// 仓库存储的载体

private LinkedList<Object> list = new LinkedList<Object>();

// 锁

private final Lock lock = new ReentrantLock();

// 仓库满的条件变量

private final Condition full = lock.newCondition();

// 仓库空的条件变量

private final Condition empty = lock.newCondition();

// 生产num个产品

public void produce(int num)

{

// 获得锁

lock.lock();

// 如果仓库剩余容量不足

while (list.size() + num > MAX\_SIZE)

{

System.out.println("【要生产的产品数量】:" + num + "/t【库存量】:" + list.size()

+ "/t暂时不能执行生产任务!");

try

{

// 由于条件不满足，生产阻塞

full.await();

}

catch (InterruptedException e)

{

e.printStackTrace();

}

}

// 生产条件满足情况下，生产num个产品

for (int i = 1; i <= num; ++i)

{

list.add(new Object());

}

System.out.println("【已经生产产品数】:" + num + "/t【现仓储量为】:" + list.size());

// 唤醒其他所有线程

full.signalAll();

empty.signalAll();

// 释放锁

lock.unlock();

}

// 消费num个产品

public void consume(int num)

{

// 获得锁

lock.lock();

// 如果仓库存储量不足

while (list.size() < num)

{

System.out.println("【要消费的产品数量】:" + num + "/t【库存量】:" + list.size()

+ "/t暂时不能执行生产任务!");

try

{

// 由于条件不满足，消费阻塞

empty.await();

}

catch (InterruptedException e)

{

e.printStackTrace();

}

}

// 消费条件满足情况下，消费num个产品

for (int i = 1; i <= num; ++i)

{

list.remove();

}

System.out.println("【已经消费产品数】:" + num + "/t【现仓储量为】:" + list.size());

// 唤醒其他所有线程

full.signalAll();

empty.signalAll();

// 释放锁

lock.unlock();

}

// set/get方法

public int getMAX\_SIZE()

{

return MAX\_SIZE;

}

public LinkedList<Object> getList()

{

return list;

}

public void setList(LinkedList<Object> list)

{

this.list = list;

}

}

【要消费的产品数量】:50 【库存量】:0 暂时不能执行生产任务!

【要消费的产品数量】:30 【库存量】:0 暂时不能执行生产任务!

【已经生产产品数】:10 【现仓储量为】:10

【已经生产产品数】:10 【现仓储量为】:20

【要消费的产品数量】:50 【库存量】:20 暂时不能执行生产任务!

【要消费的产品数量】:30 【库存量】:20 暂时不能执行生产任务!

【已经生产产品数】:10 【现仓储量为】:30

【要消费的产品数量】:50 【库存量】:30 暂时不能执行生产任务!

【已经消费产品数】:20 【现仓储量为】:10

【已经生产产品数】:10 【现仓储量为】:20

【要消费的产品数量】:30 【库存量】:20 暂时不能执行生产任务!

【已经生产产品数】:80 【现仓储量为】:100

【要生产的产品数量】:10 【库存量】:100 暂时不能执行生产任务!

【已经消费产品数】:50 【现仓储量为】:50

【已经生产产品数】:10 【现仓储量为】:60

【已经消费产品数】:30 【现仓储量为】:30

【已经生产产品数】:10 【现仓储量为】:40

import java.util.concurrent.LinkedBlockingQueue;

/\*\*

\* 仓库类Storage实现缓冲区

\*

\* Email:530025983@qq.com

\*

\* @author MONKEY.D.MENG 2011-03-15

\*

\*/

public class Storage

{

// 仓库最大存储量

private final int MAX\_SIZE = 100;

// 仓库存储的载体

private LinkedBlockingQueue<Object> list = new LinkedBlockingQueue<Object>(

100);

// 生产num个产品

public void produce(int num)

{

// 如果仓库剩余容量为0

if (list.size() == MAX\_SIZE)

{

System.out.println("【库存量】:" + MAX\_SIZE + "/t暂时不能执行生产任务!");

}

// 生产条件满足情况下，生产num个产品

for (int i = 1; i <= num; ++i)

{

try

{

// 放入产品，自动阻塞

list.put(new Object());

}

catch (InterruptedException e)

{

e.printStackTrace();

}

System.out.println("【现仓储量为】:" + list.size());

}

}

// 消费num个产品

public void consume(int num)

{

// 如果仓库存储量不足

if (list.size() == 0)

{

System.out.println("【库存量】:0/t暂时不能执行生产任务!");

}

// 消费条件满足情况下，消费num个产品

for (int i = 1; i <= num; ++i)

{

try

{

// 消费产品，自动阻塞

list.take();

}

catch (InterruptedException e)

{

e.printStackTrace();

}

}

System.out.println("【现仓储量为】:" + list.size());

}

// set/get方法

public LinkedBlockingQueue<Object> getList()

{

return list;

}

public void setList(LinkedBlockingQueue<Object> list)

{

this.list = list;

}

public int getMAX\_SIZE()

{

return MAX\_SIZE;

}

}

【库存量】:0 暂时不能执行生产任务!

【库存量】:0 暂时不能执行生产任务!

【现仓储量为】:1

【现仓储量为】:1

【现仓储量为】:3

【现仓储量为】:4

【现仓储量为】:5

【现仓储量为】:6

【现仓储量为】:7

【现仓储量为】:8

【现仓储量为】:9

【现仓储量为】:10

【现仓储量为】:11

【现仓储量为】:1

【现仓储量为】:2

【现仓储量为】:13

【现仓储量为】:14

【现仓储量为】:17

【现仓储量为】:19

【现仓储量为】:20

【现仓储量为】:21

【现仓储量为】:22

【现仓储量为】:23

【现仓储量为】:24

【现仓储量为】:25

【现仓储量为】:26

【现仓储量为】:12

【现仓储量为】:1

【现仓储量为】:1

【现仓储量为】:2

【现仓储量为】:3

【现仓储量为】:4

【现仓储量为】:5

【现仓储量为】:6

【现仓储量为】:7

【现仓储量为】:27

【现仓储量为】:8

【现仓储量为】:6

【现仓储量为】:18

【现仓储量为】:2

【现仓储量为】:3

【现仓储量为】:4

【现仓储量为】:5

【现仓储量为】:6

【现仓储量为】:7

【现仓储量为】:8

【现仓储量为】:9

【现仓储量为】:10

【现仓储量为】:16

【现仓储量为】:11

【现仓储量为】:12

【现仓储量为】:13

【现仓储量为】:14

【现仓储量为】:15

【现仓储量为】:1

【现仓储量为】:2

【现仓储量为】:3

【现仓储量为】:3

【现仓储量为】:15

【现仓储量为】:1

【现仓储量为】:0

【现仓储量为】:1

【现仓储量为】:1

【现仓储量为】:1

【现仓储量为】:2

【现仓储量为】:3

【现仓储量为】:4

【现仓储量为】:0

【现仓储量为】:1

【现仓储量为】:5

【现仓储量为】:6

【现仓储量为】:7

【现仓储量为】:8

【现仓储量为】:9

【现仓储量为】:10

【现仓储量为】:11

【现仓储量为】:12

【现仓储量为】:13

【现仓储量为】:14

【现仓储量为】:15

【现仓储量为】:16

【现仓储量为】:17

【现仓储量为】:1

【现仓储量为】:1

【现仓储量为】:2

【现仓储量为】:3

【现仓储量为】:4

【现仓储量为】:5

【现仓储量为】:6

【现仓储量为】:3

【现仓储量为】:3

【现仓储量为】:1

【现仓储量为】:2

【现仓储量为】:3

【现仓储量为】:4

【现仓储量为】:5

【现仓储量为】:6

【现仓储量为】:7

【现仓储量为】:8

【现仓储量为】:9

【现仓储量为】:10

【现仓储量为】:11

【现仓储量为】:12

【现仓储量为】:13

【现仓储量为】:14

【现仓储量为】:15

【现仓储量为】:16

【现仓储量为】:17

【现仓储量为】:18

【现仓储量为】:19

【现仓储量为】:6

【现仓储量为】:7

【现仓储量为】:8

【现仓储量为】:9

【现仓储量为】:10

【现仓储量为】:11

【现仓储量为】:12

【现仓储量为】:13

【现仓储量为】:14

【现仓储量为】:15

【现仓储量为】:16

【现仓储量为】:17

【现仓储量为】:18

【现仓储量为】:19

【现仓储量为】:20

【现仓储量为】:21

【现仓储量为】:22

【现仓储量为】:23

【现仓储量为】:24

【现仓储量为】:25

【现仓储量为】:26

【现仓储量为】:27

【现仓储量为】:28

【现仓储量为】:29

【现仓储量为】:30

【现仓储量为】:31

【现仓储量为】:32

【现仓储量为】:33

【现仓储量为】:34

【现仓储量为】:35

【现仓储量为】:36

【现仓储量为】:37

【现仓储量为】:38

【现仓储量为】:39

【现仓储量为】:40