**Q1. In this challenge, a banking system. Create the Account and Transaction classes.**

1. **The Account class has a data member int balance, initially assigned to zero. The class should implement the following three methods:**

**1.diposite()**

**2.withdrow()**

**3.getbalance()**

**Answer ::**

**import java.security.SecureRandom;**

**import java.util.List;**

**import java.util.Scanner;**

**//---- My code ----**

**import java.util.ArrayList;**

**/\***

**\* Create the Account and Transaction classes here.**

**\*/**

**class Account {**

**int balance=0;**

**public String deposit(int money){**

**balance += money;**

**return "Depositing $"+money;**

**}**

**public String withdraw(int money){**

**if(balance<money){**

**return "Withdraw $"+money+" (Insufficient Balance)";**

**} else{**

**balance -= money;**

**return "Withdraw $"+money;**

**}**

**}**

**public int getBalance(){**

**return balance;**

**}**

**}**

**class Transaction {**

**Account account = new Account();**

**List<String> transactions = new ArrayList<>();**

**public Transaction(Account account){**

**this.account = account;**

**}**

**public void deposit(int money){**

**transactions.add(account.deposit(money));**

**}**

**public void withdraw(int money){**

**transactions.add(account.withdraw(money));**

**}**

**public List<String> getTransaction(){**

**return transactions;**

**}**

**}**

**//---- End my code ----**

**public class Solution {**

**private static final Scanner SCANNER = new Scanner(System.in);**

**private static final Account ACCOUNT = new Account();**

**private static final Transaction TRANSACTION =** new **Transaction(ACCOUNT);**

**public static void main(String[] args) throws InterruptedException {**

**Comment on gist**

**Notifications for this thread**

**int threadsCount = Integer.parseInt(SCANNER.nextLine());**

**Thread[] threads = new Thread[threadsCount];**

**int expectedTransactionsCount = 0;**

**for (int i = 0; i < threadsCount; i++) {**

**int transactions Count = Integer.parseInt(SCANNER.nextLine());**

**expectedTransactionsCount += transactionsCount;**

**threads[i] = new Thread(new TransactionRunnable(TRANSACTION, transactionsCount));**

**}**

**for (int i = 0; i < threadsCount; i++) {**

**threads[i].start();**

**}**

**for (int i = 0; i < threadsCount; i++) {**

**threads[i].join();**

**}**

**List<String> transactions = TRANSACTION.getTransaction();**

**if (transactions.size() != expectedTransactionsCount) {**

**System.out.println("Wrong Answer");**

**} else {**

**boolean correct = true;**

**for (String transaction: transactions) {**

**if (transaction == null) {**

**correct = false;**

**break;**

**}**

**}**

**if (!correct) {**

**System.out.println("Wrong Answer");**

**} else {**

**int balance = ACCOUNT.getBalance();**

**if (balance < 0) {**

**System.out.println("Wrong Answer");**

**} else {**

**for (String transaction: transactions) {**

**System.out.println(transaction);**

**}**

**System.out.println("Balance $" + balance);**

**}**

**}**

**}**

**}**

**}**

**Q2. Write a program of producer and consumer.**

**import java.util.LinkedList;**

**public class Threadexample {**

**public static void main(String[] args)**

**throws InterruptedException**

**// Object of a class that has both produce()**

**// and consume() methods**

**final PC pc = new PC();**

**// Create producer thread**

**Thread t1 = new Thread(new Runnable() {**

**@Override**

**public void run()**

**{**

**try {**

**pc.consume();**

**}**

**catch (InterruptedException e) {**

**e.printStackTrace();**

**}**

**}**

**});**

**// Start both threads**

**t1.start();**

**t2.start();**

**// t1 finishes before t2**

**t1.join();**

**t2.join();**

**// This class has a list, producer (adds items to list**

**// and consumber (removes items).**

**public static class PC {**

**// Create a list shared by producer and consumer**

**try {**

**pc.consume();**

**}**

**catch (InterruptedException e) {**

**e.printStackTrace();**

**}**

**}**

**});**

**// Start both threads**

**t1.start();**

**t2.start();**

**// t1 finishes before t2**

**t1.join();**

**t2.join();**

**// This class has a list, producer (adds items to list**

**// and consumber (removes items).**

**public static class PC {**

**// Create a list shared by producer and consumer**

**// Size of list is 2.**

**LinkedList<Integer> list = new LinkedList<>();**

**int capacity = 2;**

**// Function called by producer thread**

**public void produce() throws InterruptedException**

**{**

**int value = 0;**

**while (true) {**

**synchronized (this)**

**{**

**// producer thread waits while list**

**// is full**

**while (list.size() == capacity)**

**wait();**

**System.out.println("Producer produced-"**

**+ value);**

**// to insert the jobs in the list**

**list.add(value++);**

**// notifies the consumer thread that**

**// now it can start consuming**

**notify();**

**// makes the working of program easier**

**// to  understand**

**Thread.sleep(1000);**

**}**

**}**

**}**

**// Function called by consumer thread**

**public void consume() throws InterruptedException**

**{**

**while (true) {**

**synchronized (this)**

**{**

**// consumer thread waits while list**

**// is empty**

**while (list.size() == 0)**

**wait();**

**// to retrive the ifrst job in the list**

**int val = list.removeFirst();**

**System.out.println("Consumer consumed-"**

**+ val);**

**// Wake up producer t**

**Notify();**

**// and sleep**

**Thread.sleep(1000);**

**}**

**}**

**}**

**}**

**Output:**

**Producer produced-0**

**Producer produced-1**

**Consumer consumed-0**

**Consumer consumed-1**

**Producer produced-**

**Q3. Write a program of thread to used all the method of thread like (wait, sleep, notify, notifyall,join,getname, currentthreade islive?, priority ).**

**You have to write on signle program to used**

**all.**

**Answer:**

**Class A extends Thread {**

**Public void run()**

**{**

**For (int I = 1; I <= 4; i++) {**

**Try {**

**Wait();**

**}**

**Catch (Exception e) {**

**System.out.println€;**

**}**

**System.out.println(I + “ “);**

**}**

**}**

**}**

**Class B extends Thread {**

**Public void run()**

**{ System.out.println(Thread.currentThread().getName() + “ in control”);**

**For (char I = ‘a’; I <= ‘d’; i++) {**

**Try {**

**Thread.sleep(100);**

**}**

**Catch (Exception e) {**

**System.out.println€;**

**}**

**System.out.println(I + “ “);**

**}**

**}**

**}**

**Class allmethod extends Thread {**

**Public static void main(String args[])**

**{**

**// creating two threads**

**A a1 = new A();**

**B b1 = new B();**

**// starts second thread after when**

**// first thread a1 is died.**

**A1.start();**

**A1.setPriority(5);**

**System.out.println(a1.isAlive());**

**B1.start();**

**Try {**

**A1.join();**

**}**

**Catch (Exception e) {**

**System.out.println€;**

**}**

**}**

**}**

**. Q4. You are required to compute the power of a number by implementing a calculator. Create a class MyCalculator which consists of a single method long power(int, int). This method takes two integers, n and p, as parameters and finds . If either p or n is negative, then the method must throw an exception which says "n and p should not be negative". Also, if both and are zero, then the method must throw an exception which says "n and p should not be zero"**

**Answer::**

**Import java.util.Scanner;**

**Class Mycomputer{**

**Public static int power**

**import java.util.Scanner;**

**class MyCalculator {**

**public static int power(int n,int p) throws Exception {**

**if(n<0 || p<0)**

**{**

**throw new Exception ("n or p should not be negative.");**

**}**

**else if(n==0 && p==0)**

**{**

**throw new Exception ("n and p should not be zero.");**

**}**

**else{return((int)Math.pow(n,p));**

**}**

**}**

**}**

**public class Solution**

**{**

**public static final MyCalculator my\_calculator = new MyCalculator();**

**public static final Scanner in = new Scanner(System.in);**

**public static void main(String[] args)**

**{**

**while (in .hasNextInt())**

**{**

**int n = in .nextInt();**

**int p = in .nextInt();**

**try**

**{**

**System.out.println(my\_calculator.power(n, p));**

**}**

**catch (Exception e) {**

**System.out.println(e);**

**}**

**}**

**}**

**}**

**Q5.** **Java Program Showing Execution of Multiple Tasks with a Single Thread**

**Answer::**

**class Execution extends Thread{**

**Public void run(){**

**System.out.println(“Task one”);**

**For(int i=1;i<11;i++)**

**{**

**System.out.println(i);**

**}**

**System.out.println(“Task two”);**

**For(int j=11;j<21;j++)**

**{**

**System.out.println(j);**

**}**

**System.out.println(“Task three”);**

**For(int k=21;k<31;k++)**

**{**

**System.out.println(k);**

**}**

**}**

**Public static void main(String args[]){**

**Execution e=new E();**

**T1.start();**

**}**

**}**

**Q6.Java Program Showing Two Threads Working Simultaneously Upon Two Objects**

**class ThreadDemo extends Thread {**

**private Thread t;**

**private String threadName;**

**ThreadDemo( String name) {**

**threadName = name;**

**System.out.println("Creating " + threadName );**

**}**

**public void run() {**

**System.out.println("Running " + threadName );**

**try {**

**for(int i = 4; i > 0; i--) {**

**System.out.println("Thread: " + threadName + ", " + i);**

**// Let the thread sleep for a while.**

**Thread.sleep(50);**

**}**

**}**

**catch (InterruptedException e)**

**{**

**System.out.println("Thread " + threadName + " interrupted.");**

**}**

**System.out.println("Thread " + threadName + " exiting.");**

**}**

**public void start () {**

**System.out.println("Starting " + threadName );**

**if (t == null) {**

**t = new Thread (this, threadName);**

**t.start ();**

**}**

**}**

**}**

**public class TestThread {**

**public static void main(String args[]) {**

**ThreadDemo T1 = new ThreadDemo( "Thread-1");**

**T1.start();**

**ThreadDemo T2 = new ThreadDemo( "Thread-2");**

**T2.start();**

**}**

**}**

**Output::**

**Creating Thread-1**

**Starting Thread-1**

**Creating Thread-2**

**Starting Thread-2**

**Running Thread-1**

**Thread: Thread-1, 4**

**Running Thread-2**

**Thread: Thread-2, 4**

**Thread: Thread-1, 3**

**Thread: Thread-2, 3**

**Thread: Thread-1, 2**

**Thread: Thread-2, 2**

**Thread: Thread-1, 1**

**Thread: Thread-2, 1**

**Thread Thread-1 exiting.**

**Thread Thread-2 exiting.**

**Q7.Java Program Showing Two Threads Acting Upon a Single Object**

**Answer::**

**public class PrintTest {**

**public static void main(String[] args) {**

**PrintNumbers b = new PrintNumbers();**

**One firstThread = new One(b);**

**Two secondThread = new Two(b);**

**Three thirdThread = new Three(b);**

**secondThread.setName("second: ");**

**thirdThread.setName("third: ");**

**firstThread.setName("first: ");**

**firstThread.start();**

**secondThread.start();**

**thirdThread.start();**

**}**

**}**

**o/p:**

**first: 1**

**second: 2**

**third: 3**

**first: 11**

**second: 22**

**third: 33**

**first: 111**

**second: 222**

**third: 333**

**...and so on**

**Q8.Java Program with 2 Threads Which Prints Alternatively**

**Answer::**

**package multi\_threading;**

**public class inter\_thread implements Runnable {**

**static inter\_thread obj;**

**boolean val=false;ob**

**Thread t;**

**public inter\_thread()**

**{**

**}**

**public inter\_thread(String msg)**

**{**

**t=new Thread(obj,msg)**

**t.start();**

**}**

**public static void main(String args[]){**

**obj=new inter\_thread();**

**inter\_thread obj1=new inter\_thread("Child1");**

**inter\_thread obj2=new inter\_thread("Child2");**

**try{**

**obj1.t.join();**

**obj2.t.join();**

**}**

**catch(InterruptedException e){**

**System.out.println("Interrupted");**

**}**

**}**

**public void run(){**

**int i;**

**synchronized(obj)**

**{**

**for(i=1;i<=5;i++)**

**{**

**System.out.println(i);**

**val=!val;**

**while(val)**

**try{**

**wait();**

**}**

**catch(InterruptedException e){**

**System.out.println("Interrupted");**

**}**

**notify();**

**}**

**}**

**}**

**}**