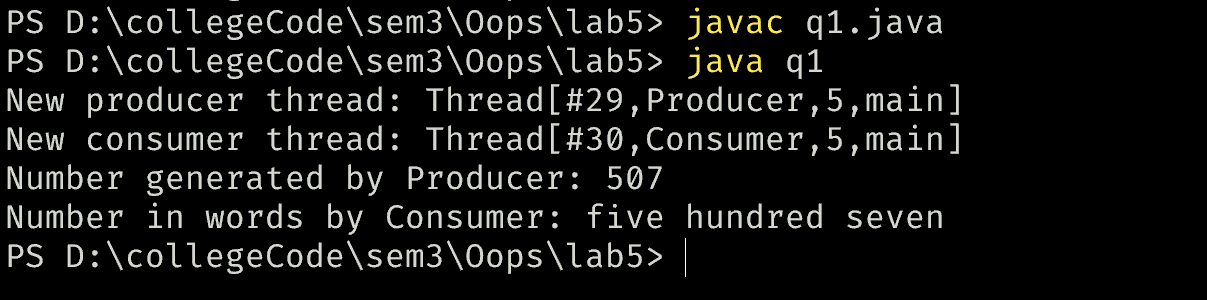
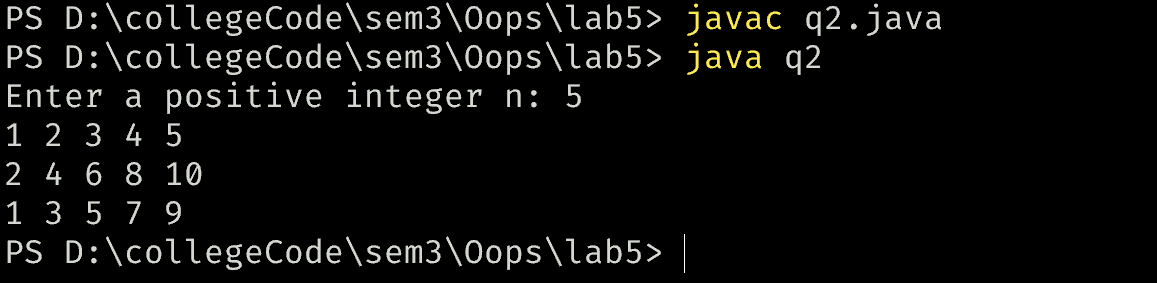
Q1  
import java.util.Random;  
  
class newProducerThread implements Runnable {  
 String name;  
 Thread t;  
 int n;  
 boolean numberGenerated = false;  
  
 newProducerThread(String threadName) {  
 this.name = threadName;  
 t = new Thread(this, name);  
 System.*out*.println("New producer thread: " + t);  
 t.start();  
 }  
  
 public void run() {  
 Random random = new Random();  
 n = random.nextInt(1000); // Generates a number between 0 and 999  
 System.*out*.println("Number generated by Producer: " + n);  
 numberGenerated = true;  
  
 try {  
 Thread.*sleep*(1000);  
 } catch (InterruptedException e) {  
 System.*out*.println("Producer thread interrupted.");  
 }  
 }  
  
 public int getNumber() {  
 return n;  
 }  
  
 public boolean isNumberGenerated() {  
 return numberGenerated;  
 }  
}  
  
class newConsumerThread implements Runnable {  
 String name;  
 Thread t;  
 newProducerThread producer;  
  
 newConsumerThread(String threadName, newProducerThread producer) {  
 this.name = threadName;  
 this.producer = producer;  
 t = new Thread(this, name);  
 System.*out*.println("New consumer thread: " + t);  
 t.start();  
 }  
  
 public void run() {  
 while (!producer.isNumberGenerated()) {  
 try {  
 Thread.*sleep*(1000);  
 } catch (InterruptedException e) {  
 System.*out*.println("Consumer thread interrupted.");  
 }  
 }  
  
 int number = producer.getNumber();  
 System.*out*.println("Number in words by Consumer: " + numberToWords(number));  
 }  
  
 private String numberToWords(int number) {  
 if (number == 0) return "zero";  
  
 String[] units = {"", "one", "two", "three", "four", "five", "six", "seven", "eight", "nine"};  
 String[] teens = {"", "eleven", "twelve", "thirteen", "fourteen", "fifteen", "sixteen", "seventeen", "eighteen", "nineteen"};  
 String[] tens = {"", "ten", "twenty", "thirty", "forty", "fifty", "sixty", "seventy", "eighty", "ninety"};  
 String[] hundreds = {"", "one hundred", "two hundred", "three hundred", "four hundred", "five hundred", "six hundred", "seven hundred", "eight hundred", "nine hundred"};  
  
 StringBuilder words = new StringBuilder();  
  
 if (number >= 100) {  
 words.append(hundreds[number / 100]).append(" ");  
 number %= 100;  
 }  
  
 if (number >= 20) {  
 words.append(tens[number / 10]).append(" ");  
 number %= 10;  
 } else if (number > 10) {  
 words.append(teens[number - 10]).append(" ");  
 number = 0;  
 } else if (number == 10) {  
 words.append(tens[1]).append(" ");  
 number = 0;  
 }  
  
 if (number > 0) {  
 words.append(units[number]);  
 }  
  
 return words.toString().trim();  
 }  
}  
  
public class q1 {  
 public static void main(String[] args) {  
 newProducerThread producer = new newProducerThread("Producer");  
 newConsumerThread consumer = new newConsumerThread("Consumer", producer);  
 }  
}

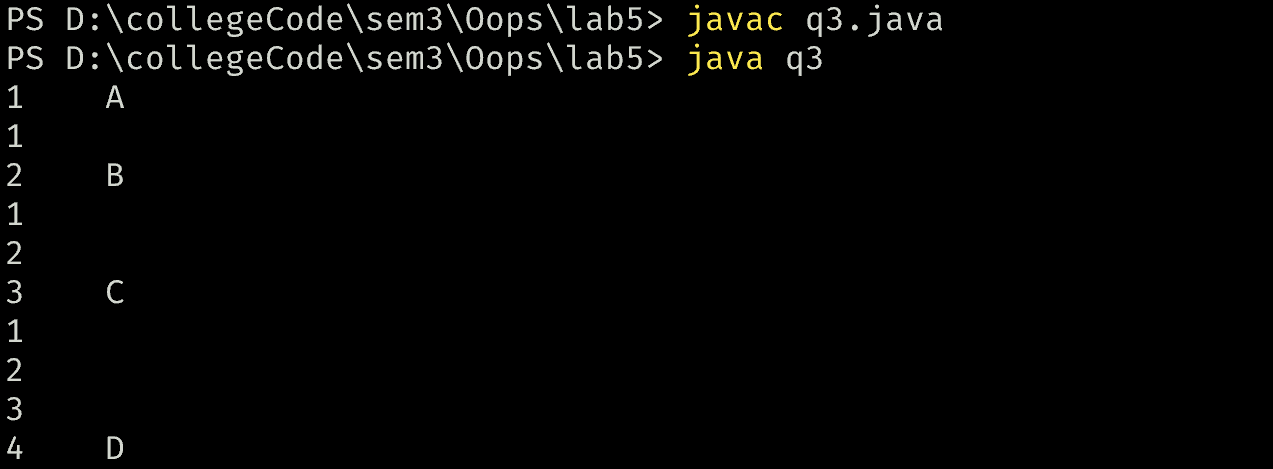


Q2

import java.util.Scanner;  
  
class NaturalNumbers implements Runnable {  
 private int n;  
  
 public NaturalNumbers(int n) {  
 this.n = n;  
 }  
  
 @Override  
 public void run() {  
 for (int i = 1; i <= n; i++) {  
 System.*out*.print(i + " ");  
 }  
 System.*out*.println();  
 }  
}  
  
class EvenNumbers implements Runnable {  
 private int n;  
  
 public EvenNumbers(int n) {  
 this.n = n;  
 }  
  
 @Override  
 public void run() {  
 for (int i = 1; i <= n; i++) {  
 System.*out*.print((i \* 2) + " ");  
 }  
 System.*out*.println();  
 }  
}  
  
class OddNumbers implements Runnable {  
 private int n;  
  
 public OddNumbers(int n) {  
 this.n = n;  
 }  
  
 @Override  
 public void run() {  
 for (int i = 1; i <= n; i++) {  
 System.*out*.print((i \* 2 - 1) + " ");  
 }  
 System.*out*.println();  
 }  
}  
  
public class q2 {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 System.*out*.print("Enter a positive integer n: ");  
 int n = scanner.nextInt(); // User input for n  
  
 // Input validation  
 if (n <= 0) {  
 System.*out*.println("Please enter a positive integer.");  
 scanner.close();  
 return;  
 }  
  
 Thread naturalThread = new Thread(new NaturalNumbers(n));  
 Thread evenThread = new Thread(new EvenNumbers(n));  
 Thread oddThread = new Thread(new OddNumbers(n));  
  
 naturalThread.start();  
 try {  
 naturalThread.join(); // Wait for the natural numbers thread to finish  
 } catch (InterruptedException e) {  
 Thread.*currentThread*().interrupt();  
 }  
  
 evenThread.start();  
 try {  
 evenThread.join(); // Wait for the even numbers thread to finish  
 } catch (InterruptedException e) {  
 Thread.*currentThread*().interrupt();  
 }  
  
 oddThread.start();  
 try {  
 oddThread.join(); // Wait for the odd numbers thread to finish  
 } catch (InterruptedException e) {  
 Thread.*currentThread*().interrupt();  
 }  
  
 scanner.close(); // Close the scanner  
 }  
}



Q3  
  
public class q3{  
 public static void main(String[] args) {  
 int n = 4;  
 Share share = new Share();  
 Numbers numbers = new Numbers(n, share);  
 NumberAndAlphabets numberAndAlphabets = new NumberAndAlphabets(n, share);  
 Thread t1 = new Thread(numbers);  
 Thread t2 = new Thread(numberAndAlphabets);  
 t1.start();  
 t2.start();  
 }  
}  
class Share{  
 private boolean flag = true;  
  
 public synchronized void printNumber(int n){  
 while(!flag){  
 try{  
 wait();  
 }catch(InterruptedException e){  
 System.*out*.println("Thread interrupted"+e);  
 }  
 }  
 for (int i = 1; i < n; i++) {  
 System.*out*.println(i);  
 }  
 flag = false;  
 notifyAll();  
 }  
 public synchronized void printAlphabet(int n){  
 char alphabet = (char) ('A' + (n - 1));  
 while(flag){  
 try{  
 wait();  
 }catch(InterruptedException e){  
 System.*out*.println("Thread interrupted"+e);  
 }  
 }  
 System.*out*.println(n + " " + alphabet);  
 flag = true;  
 notifyAll();  
}  
}  
class NumberAndAlphabets implements Runnable{  
 private int n;  
 private Share share;  
  
 public NumberAndAlphabets(int n , Share share){  
 this.n = n;  
 this.share = share;  
 }  
  
 @Override  
 public void run(){  
 for(int i = 1 ; i <= n ; i++){  
 share.printAlphabet(i);  
 }  
 }  
}  
  
class Numbers implements Runnable{  
 private int n;   
 private Share share;  
  
 public Numbers(int n, Share share){  
 this.n = n;  
 this.share = share;  
 }  
 @Override  
 public void run(){  
 for(int i = 1; i <= n ; i++){  
 share.printNumber(i);  
 }  
 }  
}



Q4

import java.util.Scanner;  
  
class SleepThread extends Thread {  
 private long sleepTime;  
 private String threadName;  
  
 public SleepThread(long sleepTime, String threadName) {  
 this.sleepTime = sleepTime;  
 this.threadName = threadName;  
 }  
  
 @Override  
 public void run() {  
 try {  
 System.*out*.println(threadName + ": Sleeping for " + sleepTime + " milliseconds.");  
 Thread.*sleep*(sleepTime);  
 System.*out*.println(threadName + ": Woke up after " + sleepTime + " milliseconds.");  
 } catch (InterruptedException e) {  
 Thread.*currentThread*().interrupt();  
 System.*out*.println(threadName + ": Interrupted.");  
 }  
 }  
}  
  
public class q4 {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
  
 System.*out*.print("Enter the sleep time for the main thread (x in milliseconds): ");  
 long x = scanner.nextLong();  
  
 System.*out*.print("Enter the sleep time for T1 (y in milliseconds): ");  
 long y = scanner.nextLong();  
  
 System.*out*.print("Enter the sleep time for T2 (z in milliseconds): ");  
 long z = scanner.nextLong();  
  
 // Create threads  
 SleepThread t1 = new SleepThread(y, "T1");  
 SleepThread t2 = new SleepThread(z, "T2");  
  
 // Main thread behavior  
 try {  
 System.*out*.println("Main Thread: Sleeping for " + x + " milliseconds.");  
 Thread.*sleep*(x);  
 System.*out*.println("Main Thread: Woke up after " + x + " milliseconds.");  
 } catch (InterruptedException e) {  
 Thread.*currentThread*().interrupt();  
 System.*out*.println("Main Thread: Interrupted.");  
 }  
  
 // Start T1 and wait for it to finish  
 t1.start();  
 try {  
 t1.join(); // Wait for T1 to finish  
 } catch (InterruptedException e) {  
 Thread.*currentThread*().interrupt();  
 System.*out*.println("Main Thread: Interrupted while waiting for T1.");  
 }  
  
 // Start T2 and wait for it to finish  
 t2.start();  
 try {  
 t2.join(); // Wait for T2 to finish  
 } catch (InterruptedException e) {  
 Thread.*currentThread*().interrupt();  
 System.*out*.println("Main Thread: Interrupted while waiting for T2.");  
 }  
  
 // Close the scanner  
 scanner.close();  
 }  
}

