

Homework 5

Due date: Sunday, March 27, 2022, 11:59pm

Instructions:

- For each day of late submission, 10 points will be deducted.
- After two days, no more submission is allowed.
- You must submit through Canvas.
- Keep this file intact, copy your solution codes from your IDE and paste at the bottom of the file under 'Answer'. Submit this word document with your codes. Do not change the format of the file to pdf or .rar or anything else.

Reminder: [Academic Integrity](#) policy is strictly implemented on all your submissions.

Learning Objectives:

- Create thread life cycle and thread priority & thread scheduling.
- Illustrate synchronization and context switching in Java multithreaded program.

Write a multithreaded java program where a person gains 1000 calories throughout the day. But, the person burns it off too. So, the balance is zero. That means no gain or loss in calories. Synchronization will enable you to achieve zero balance every time the program is run.

Note: First, do not think about synchronization. Write the program with two threads, one java file can handle the task – I named it Zerosum. There are two functions here, gainCalories() and burnCalories(). Each thread handles one of those functions. The run() method of each thread will have the method details. You can also call those methods if you decide to develop it outside the thread block. You are keeping a count of 0 to 1000 calories for both gain and loss. You need to keep track of the balance, so that is your instance variable which you print at end of the program. At the beginning of the program balance is zero.

When you run this multithreaded program without synchronization, each run will show a result - a single integer. It can be positive integer, negative integer and occasionally 0. It's the value of 'balance'. It's interleaved. That means both the threads are running but not in an orderly fashion.

Then you synchronize and run the program. Each time the balance will be 0. That is the output you want to have.

Name your file Zerosum.java

Answer (synchronized):

<paste code here for Zerosum.java>

```
import java.lang.Thread;
import java.lang.Runnable;

public class Zerosum implements Runnable {

    private int balance ,count, origin;

    // create gainCalories function

    public void gainCalories() {
        for (int i=1; i<1001; i++) {
            balance += count;
            count++;
            System.out.print(count + ", ");
        }
    }

    //create burnCalories function

    public void burnCalories() {
        for (int i=1; i<1001; i++) {
            origin = 1000 - count;
            balance -= origin;
            count--;
            System.out.print(count + ", ");
        }
    }

    //implementing run method to run thread and synchronizing it

    public synchronized void run() {
        gainCalories();
        System.out.println();
        burnCalories();
        System.out.println();
    }

    // implement main method. So the objects of thread will start

    public static void main(String[] args) {
        Zerosum obj = new Zerosum();
        Thread td1 = new Thread(obj);
        Thread td2 = new Thread(obj);
```

```

//starting the threads

td1.start();
td2.start();

try {
td1.join();
td2.join();
} catch (Exception e) {
    System.out.println("Error");
}

System.out.println();
System.out.println("Final Result of Balance: " + obj.balance);
}
}

```

- Screenshot of output:

The screenshot shows an IDE with a Java project named 'Homework5'. The source file 'Zerosum.java' is open, displaying the following code:

```

29 //implementing run method to run thread and synchronizing it
30
31
32 public synchronized void run() {
33     gainCalories();
34     System.out.println();
35     burnCalories();
36     System.out.println();
37 }
38
39 // implement main method. So the objects of thread will start
40
41 public static void main(String[] args) {
42     Zerosum obj = new Zerosum();
43     Thread td1 = new Thread(obj);
44     Thread td2 = new Thread(obj);
45
46     //starting the threads
47
48     td1.start();
49     td2.start();
50
51     try {
52         td1.join();
53         td2.join();
54     } catch (Exception e) {
55         System.out.println("Error");
56     }
57
58     System.out.println();
59     System.out.println("Final Result of Balance: " + obj.balance);
60 }
61 }

```

The console output shows the execution of the program, displaying a long sequence of memory addresses (e.g., 999, 998, 997, 996, 995, 994, 993, 992, 991, 990, 989, 988, 987, 986, 985, 984, 983, 982, 981, 980, 979, 978, 977, 976, 975, 974, 973, 972, 971, 970, 969, 968, 967, 966, 965, 964, 963, 962, 961, 960, 959, 958, 957, 956, 955, 954, 953, 952, 951, 950, 949, 948, 947, 946, 945, 944, 943, 942, 941, 940, 939, 938, 937, 936, 935, 934, 933, 932, 931, 930, 929, 928, 927, 926, 925, 924, 923, 922, 921, 920, 919, 918, 917, 916, 915, 914, 913, 912, 911, 910, 909, 908, 907, 906, 905, 904, 903, 902, 901, 900, 899, 898, 897, 896, 895, 894, 893, 892, 891, 890, 889, 888, 887, 886, 885, 884, 883, 882, 881, 880, 879, 878, 877, 876, 875, 874, 873, 872, 871, 870, 869, 868, 867, 866, 865, 864, 863, 862, 861, 860, 859, 858, 857, 856, 855, 854, 853, 852, 851, 850, 849, 848, 847, 846, 845, 844, 843, 842, 841, 840, 839, 838, 837, 836, 835, 834, 833, 832, 831, 830, 829, 828, 827, 826, 825, 824, 823, 822, 821, 820, 819, 818, 817, 816, 815, 814, 813, 812, 811, 810, 809, 808, 807, 806, 805, 804, 803, 802, 801, 800, 799, 798, 797, 796, 795, 794, 793, 792, 791, 790, 789, 788, 787, 786, 785, 784, 783, 782, 781, 780, 779, 778, 777, 776, 775, 774, 773, 772, 771, 770, 769, 768, 767, 766, 765, 764, 763, 762, 761, 760, 759, 758, 757, 756, 755, 754, 753, 752, 751, 750, 749, 748, 747, 746, 745, 744, 743, 742, 741, 740, 739, 738, 737, 736, 735, 734, 733, 732, 731, 730, 729, 728, 727, 726, 725, 724, 723, 722, 721, 720, 719, 718, 717, 716, 715, 714, 713, 712, 711, 710, 709, 708, 707, 706, 705, 704, 703, 702, 701, 700, 699, 698, 697, 696, 695, 694, 693, 692, 691, 690, 689, 688, 687, 686, 685, 684, 683, 682, 681, 680, 679, 678, 677, 676, 675, 674, 673, 672, 671, 670, 669, 668, 667, 666, 665, 664, 663, 662, 661, 660, 659, 658, 657, 656, 655, 654, 653, 652, 651, 650, 649, 648, 647, 646, 645, 644, 643, 642, 641, 640, 639, 638, 637, 636, 635, 634, 633, 632, 631, 630, 629, 628, 627, 626, 625, 624, 623, 622, 621, 620, 619, 618, 617, 616, 615, 614, 613, 612, 611, 610, 609, 608, 607, 606, 605, 604, 603, 602, 601, 600, 599, 598, 597, 596, 595, 594, 593, 592, 591, 590, 589, 588, 587, 586, 585, 584, 583, 582, 581, 580, 579, 578, 577, 576, 575, 574, 573, 572, 571, 570, 569, 568, 567, 566, 565, 564, 563, 562, 561, 560, 559, 558, 557, 556, 555, 554, 553, 552, 551, 550, 549, 548, 547, 546, 545, 544, 543, 542, 541, 540, 539, 538, 537, 536, 535, 534, 533, 532, 531, 530, 529, 528, 527, 526, 525, 524, 523, 522, 521, 520, 519, 518, 517, 516, 515, 514, 513, 512, 511, 510, 509, 508, 507, 506, 505, 504, 503, 502, 501, 500, 499, 498, 497, 496, 495, 494, 493, 492, 491, 490, 489, 488, 487, 486, 485, 484, 483, 482, 481, 480, 479, 478, 477, 476, 475, 474, 473, 472, 471, 470, 469, 468, 467, 466, 465, 464, 463, 462, 461, 460, 459, 458, 457, 456, 455, 454, 453, 452, 451, 450, 449, 448, 447, 446, 445, 444, 443, 442, 441, 440, 439, 438, 437, 436, 435, 434, 433, 432, 431, 430, 429, 428, 427, 426, 425, 424, 423, 422, 421, 420, 419, 418, 417, 416, 415, 414, 413, 412, 411, 410, 409, 408, 407, 406, 405, 404, 403, 402, 401, 400, 399, 398, 397, 396, 395, 394, 393, 392, 391, 390, 389, 388, 387, 386, 385, 384, 383, 382, 381, 380, 379, 378, 377, 376, 375, 374, 373, 372, 371, 370, 369, 368, 367, 366, 365, 364, 363, 362, 361, 360, 359, 358, 357, 356, 355, 354, 353, 352, 351, 350, 349, 348, 347, 346, 345, 344, 343, 342, 341, 340, 339, 338, 337, 336, 335, 334, 333, 332, 331, 330, 329, 328, 327, 326, 325, 324, 323, 322, 321, 320, 319, 318, 317, 316, 315, 314, 313, 312, 311, 310, 309, 308, 307, 306, 305, 304, 303, 302, 301, 300, 299, 298, 297, 296, 295, 294, 293, 292, 291, 290, 289, 288, 287, 286, 285, 284, 283, 282, 281, 280, 279, 278, 277, 276, 275, 274, 273, 272, 271, 270, 269, 268, 267, 266, 265, 264, 263, 262, 261, 260, 259, 258, 257, 256, 255, 254, 253, 252, 251, 250, 249, 248, 247, 246, 245, 244, 243, 242, 241, 240, 239, 238, 237, 236, 235, 234, 233, 232, 231, 230, 229, 228, 227, 226, 225, 224, 223, 222, 221, 220, 219, 218, 217, 216, 215, 214, 213, 212, 211, 210, 209, 208, 207, 206, 205, 204, 203, 202, 201, 200, 199, 198, 197, 196, 195, 194, 193, 192, 191, 190, 189, 188, 187, 186, 185, 184, 183, 182, 181, 180, 179, 178, 177, 176, 175, 174, 173, 172, 171, 170, 169, 168, 167, 166, 165, 164, 163, 162, 161, 160, 159, 158, 157, 156, 155, 154, 153, 152, 151, 150, 149, 148, 147, 146, 145, 144, 143, 142, 141, 140, 139, 138, 137, 136, 135, 134, 133, 132, 131, 130, 129, 128, 127, 126, 125, 124, 123, 122, 121, 120, 119, 118, 117, 116, 115, 114, 113, 112, 111, 110, 109, 108, 107, 106, 105, 104, 103, 102, 101, 100, 99, 98, 97, 96, 95, 94, 93, 92, 91, 90, 89, 88, 87, 86, 85, 84, 83, 82, 81, 80, 79, 78, 77, 76, 75, 74, 73, 72, 71, 70, 69, 68, 67, 66, 65, 64, 63, 62, 61, 60, 59, 58, 57, 56, 55, 54, 53, 52, 51, 50, 49, 48, 47, 46, 45, 44, 43, 42, 41, 40, 39, 38, 37, 36, 35, 34, 33, 32, 31, 30, 29, 28, 27, 26, 25, 24, 23, 22, 21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0.

The final output of the program is:

```

Final Result of Balance: 0

```