```
public static void calcRealTotal (int x) {
    realTotal = (int) (Math.ceil(x / dx));
    System.out.println("realTotal = " + realTotal);

    totalRemain = realTotal % threadNum;
    xEnd = new int[threadNum];

    xStart = new int[threadNum];

    for (int i = 0; i < threadNum; i++) {
        xStart[i] = 0;
        xEnd[i] = 0;
    }
}

public void calcEachThreadTotal () {
    total = realTotal / threadNum;

    if (totalRemain-- > 0) {
        total++;
    }

    System.out.println("total = " + total);
}

public float getSum() { return sum; }
```

```
나는 솔직히 여기가 제일 어려웠음 (부모클래스 생성자도)
리얼토탈 클래스는 → 아까 얘기했듯 static으로 처리한다.
매니져클래스에서 해당 값을 사용하기 때문 (반복처리 안해도됨)
그리고 각 xStart와 xEnd값에 배열을 만든다.
(원래 안만들려고 하신것같은데, 아무래도 각 값을 "순차적으로 중복없이" 대입하고 넣고 하기엔 배열을 만드는게 더 효율적인였던 것같음)
그리고 각 xS,xE 배열값이에 0으로 초기화해줌. 오? 이건 왜 하신거지?? 그냥 초기화 해주는 과정인건가?

해당 매소드는 각 토탈이 얼마인지 계산해준다.
중요한점 = 나눈 후의 나머지는?? -> 누락될 것임.
이것땜에 if문 사용

totalRemain이 나눈값의 나머지인데, 만약 나머지가 있다면 total에 값을 하나씩 더해주면서 나머지값을 하나씩 빼는 방법이다.(천재)
```

```
결론 : 위의 값을 구한 후,
해당 매소드의 TOTAL 값으로
옆에있는
xStart 와 xEnd 값을 구해야한다는 뜻
```

```
// 1. 등분 했을 경우 스레드 각각이 둘러야 하는 전체 숫자를 먼저 파악
// 2. 이것을 베이스로 시작과 끝 값을 정하면 됨
public ThreadCommon(int x, int y) {
    this.x = x;
    this.y = y;

    // 1. realTotal값을 통해서 threadNum으로 나눴을때 몫이 얼마가 나오는지 판정한다.
    // 2. 나머지값을 판정한다.
    // 3. 각 threadId 들에게 나머지가 0이 될 때까지 1개씩 나눠준다.
    calcEachThreadTotal();

localThreadId = threadCnt++;

// xTotalEnd는 전역으로 현재 어디까지 갔는지 파악
// xEnd는 실제 local(지역) 변수로 스레드마다 각기 다르게 가지게 해야함
if (localThreadId == 0) {
    xStart[localThreadId] = 0;
    xEnd[localThreadId] = total - 1;
} else {
    // [0] [1] [2] [3] [4] [5]
    // total 334 334 333 333 333 333
    // xStart 0 334 668 1001 1334 1667
    // xEnd 333 667 1000 1333 1666 1999
    xStart[localThreadId] = xEnd[localThreadId - 1] + 1;
    xEnd[localThreadId] = xEnd[localThreadId - 1] + total;
}
sum = 0;

System.out.printf("xStart = %4d, xEnd = %4d, thread ID = %d\n",
    xStart[localThreadId], xEnd[localThreadId], localThreadId);
```