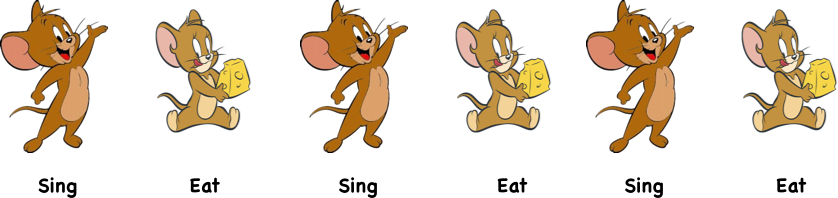
The terms *concurrency* and *parallelism* are often used in relation to multithreaded programs. At first it may seem as if concurrency and parallelism may be referring to the same concepts. Concurrent is not parallelism, and vice versa.

**Concurrency** means that an application is making progress on more than one task at the same time (concurrently). Well, if the computer only has one CPU the application may not make progress on more than one task at *exactly the same time*, but more than one task is being processed at a time inside the application. It does not completely finish one task before it begins the next.

Let's take an example in real life: There’s a challenge that requires you to both eat a whole huge cake and sing a whole song. You’ll win if you’re the fastest who sings the whole song and finishes the cake. So the rule is that you sing and eat simultaneously, How you do that does not belong to the rule. You can eat the whole cake, then sing the whole song, or you can eat half a cake, then sing half a song, then do that again, etc.



**Parallelism** means that an application splits its tasks up into smaller subtasks which can be processed in parallel, for instance on multiple CPUs at the exact same time.

If we keep going with the same example as above, the rule is still singing and eating concurrently, but this time, you play in a team of two. You probably will eat and let your friend sing (because she sings better and you eat better). So this time, the two tasks are really executed simultaneously, and it’s called *parallel*.