21.3 Dynamic Memory Allocation

Creating and maintaining dynamic data structures requires **dynamic memory allocation**—allowing a program to obtain more memory space at execution time to hold new nodes and to release space no longer needed. Remember that Java does not require you to explicitly release dynamically allocated memory. Rather, Java performs automatic *garbage collection* of objects that are no longer referenced in a program.

The limit for dynamic memory allocation can be as large as the amount of available physical memory in the computer or the amount of available disk space in a virtual-memory system. Often the limits are much smaller, because the computer's available memory must be shared among many applications.

The declaration and class-instance-creation expression

```
// 10 is nodeToAdd's data
Node<Integer> nodeToAdd = new Node<Integer>(10);
```

allocates a `Node<Integer>` object and returns a reference to it, which is assigned to `nodeToAdd`. If *insufficient memory* is available, the preceding expression throws an

`OutOfMemoryError`. The sections that follow discuss lists, stacks, queues and trees—all of which use dynamic memory allocation and self-referential classes to create dynamic data structures.