Compuer Science 62
Assignment 4

Due 11:59pm on Tuesday, February 23, 2010

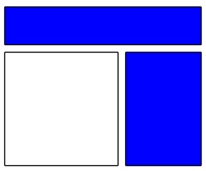
You are to write a cookie-dough allocator class. An object in the class starts with a square sheet of cookie dough. In response to requests, the object gives out squares from the sheet. As the original square is cut into smaller rectangles, the object must keep track of all the pieces. Once cut, the dough stays cut. A rectangle cannot be rejoined with other rectangles to make a larger piece.

The object maintains an inventory of rectangles of dough. When a request is made, the object chooses a suitable rectangle from its inventory. (Later, we will have more to say about how the choice is made.) The object then cuts the square from the selected rectangle and returns any remainders to the inventory. There are three possibilities:

1. The requested square is exactly the same size as the chosen rectangle. In this case, the rectangle is simply removed from the inventory.

2. The side of the requested square exactly matches one side of the rectangle. The rectangle is removed from the inventory, the square is cut out, and the remaining rectangle is returned to the inventory.

3. The side of the requested square is smaller than both dimensions of the rectangle. The rectangle is removed from the inventory, the square is cut out, and *two* smaller rectangles are returned to the inventory (see the illustration to the left). One of your implementation decisions is to decide between the two ways to divide a rectangle. We use rectangles because they are more easily stored and manipulated than more complex shapes, and there is no point in dividing the original rectangle into more than two pieces.



The assignment

You are to write a Java class, named `DoughAllocator`, that contains the following public members.

`public DoughAllocator(int size)`, a constructor which begins with a single square sheet whose side is the specified size. The constructor will fail, through a call to `assert`, if `size` is not positive or if `size` is greater than 16,000. The upper bound on `size` is chosen so that area of the square of `size` is still an ordinary integer.

`public int area()`, a method that returns the total area of dough remaining.

`public int largest()`, a method that returns the length of the **side** of the largest **square** that could currently be allocated.

`public void get(int size)`, a method that provides a square of dough of the specified size. If this were a real allocator, the method would provide actual dough, and the return type would be something other than `void`. The method will succeed if `size` is positive and not greater than the largest square available. Otherwise, the method will fail, through a call to `assert`.

In addition to the class members above, your code must use some kind of list class to maintain the inventory of rectangles. Think about what list would be appropriate here.

Please read the specifications carefully; some details are easy to overlook. For example, the call `get(0)` must not succeed, even though the request could logically be granted.

For this assignment, I have not provided any starter code. Start a new project and then create a new class named `DoughAllocator` and fill in the appropriate details. Besides the methods described above, you are free to make your own implementation decisions. Note, it may be useful to include additional classes (for example, one to represent a rectangle of dough) to assist you `DoughAllocator` class.

Dough allocation strategy

After several steps, the dough will be divided into many rectangles, and it is not obvious which rectangle to select when the next request comes in.

There is no optimum allocation strategy, because we do not know what the future requests will be. Even if we did, finding the best strategy is a very hard problem. We do, however, place a few requirements on the allocation.

- The class must grant a request for a square of size 8 immediately after constructing a sheet of size 8. This requirement eliminates the possibility of the constructor immediately dividing the dough into squares of size 1.
- Similarly, the class must grant requests for squares of size 3 and 5 immediately after constructing a sheet of size 8.
- From a newly-constructed sheet of size 8, the class must grant requests of sizes 3, 3, 3, 2, and 2. It may or may not be possible to grant a subsequent request of size 3, depending on the choices made for previous squares.
- If `area()` reports a value of 6, then it must be possible to satisfy six successive requests for squares of size 1.

Submission

Be sure to write your code carefully and test it. Be sure that your code is clear, formatted properly and commented appropriately (using Javadoc... see the first assignment for details on what is expected for comments).

Follow the directions on the course web page for submitting. Make sure if you do code supporting classes to also include those in your .jar file.

Grading

You will be graded based on the following criteria:

criterion	points
functionality/correctness	14
appropriate comments (including JavaDoc)	3
style and formatting	2
submitted correctly	1

NOTE: Code that does not compile will not be accepted! Make sure that your code compiles before submitting it.