factory method

Factories build things.

In OO parlance, a factory method returns an object of a class, building it (using a new-expression) or returning an already existing object. We explain and give examples.

Class Integer

Generally, there is no need for more than one object of class `Integer` that wraps 0, since the object is immutable. Yet, every evaluation of `new Integer(0)` creates a new object that wraps 0. Further, creating an object *does* take time. Therefore, since Java version 9, the constructors of class `Integer` have been deprecated, and this comment appears in the discussion of its constructor in the Java API:

> ... it is rarely appropriate to use this constructor. The static factory `valueOf(int)` is generally a better choice, as it is likely to yield significantly better space and time performance.

Here is part of the spec of method `valueOf(int)`:

> ... this method is likely to yield significantly better space and time performance by caching frequently requested values. This method will always cache values in the range -128 to 127 ...

How to suppress the use of a constructor and write a factory method.

First, make the constructor private! Then, only methods in the class can use the new-expression.

Second, write a public static method, with as many parameters as you need, that returns an object of the class. You get to decide, based on the parameters, whether a new object has to be created or whether an existing one can be returned.