Object Composition

You can have a situation where two different classes are related, but there is no inheritance going on. This is referred to as **composition** -- where one class makes use of code contained in another class. For example, imagine we have a **Package** class which represents a software package. It contains attributes about the software package, like name, version, and size. We also have a **Repository** class which represents all the packages available for installation. While there's no inheritance relationship between the two classes, they are related. The Repository class will contain a dictionary or list of Packages that are contained in the repository. Let's take a look at an example Repository class definition:

```
>>> class Repository:
...         def __init__(self):
...             self.packages = {}
...         def add_package(self, package):
...             self.packages[package.name] = package
...         def total_size(self):
...             result = 0
...             for package in self.packages.values():
...                 result += package.size
...             return result
```

In the constructor method, we initialize the packages dictionary, which will contain the package objects available in this repository instance. We initialize the dictionary in the constructor to ensure that every instance of the Repository class has its own dictionary.

We then define the add_package method, which takes a Package object as a parameter, and then adds it to our dictionary, using the package name attribute as the key.

Finally, we define a total_size method which computes the total size of all packages contained in our repository. This method iterates through the values in our repository dictionary and adds together the size attributes from each package object contained in the dictionary, returning the total at the end. In this example, we're making use of Package attributes within our Repository class. We're also calling the values() method on our packages dictionary instance. Composition allows us to use objects as attributes, as well as access all their attributes and methods.

Mark as completed