Pause for a Moment

Think about how you would rewrite the following code in a more object oriented way.

```
1  public class CaesarCipher {
2    public String encrypt(String input, int key) {
3      String alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
4      String shiftedAlphabet = alphabet.substring(key) +
5                               alphabet.substring(0,key);
6      StringBuilder sb = newStringBuilder(input);
7      for (int i = 0; i <sb.length(); i++) {
8        char c = sb.charAt(i);
9        int idx = alphabet.indexOf(c);
10     }
11   }
12 }
```

Continue

0:29 / 3:38

Rewriting with Encapsulation

Have a question? Discuss this lecture in the week forums.

Interactive Transcript

Search Transcript English

0:03
Welcome back. To start learning concepts of object oriented programming, it is useful to take an existing piece of code and see how you could write it differently in a more object oriented way. Here is the code you wrote in a previous lesson to perform encryption with a CaesarCipher. As you may recall one of the parameters to this method is the encryption key that you wanna use.

0:24
And the first thing you did with this code was to compute the shifted alphabet based on this key.

0:31
Here's a different way to write the CaesarCipher class, which does exactly the same thing but makes better use of the object oriented nature of Java.

0:39
This class has two fields. A field is a special kind of variable which lives inside of an object instead of inside of a method. Here, the two fields are the strings for the alphabet and the shifted alphabet and they have been moved outside of the encrypt method. Notice that they are now declared inside of the class, but outside of any method. These are now data that are encapsulated in your object. When you make a CaesarCipher object, it will have these two fields, which any code inside that class can refer to by name.

1:13
Next, notice that there is some code here which looks like a method, where we forgot to write the return type, and named it the same as the class. This code is actually a constructor, which means code that gets run to initialize an object when it is created using new.

1:31
This constructor takes in the key as a parameter and initializes the alphabet and shifted alphabet fields using the same approach you used before. If you look further down in this implementation of the CaesarCipher, the encrypt method looks mostly the same as before but it no longer takes the key as a parameter, nor does it compute the shifted alphabet. But the rest of the code is the same. In fact, the code uses the alphabet and shifted alphabet fields in the object, even though these are not declared inside the method. This code is allowed to use them because it is inside the object. So it can use any fields within the object.

2:11
An illustration helps understand the differences between these two approaches. In the old code, a CaesarCipher object held no data. When you do new CaesarCipher, you pass in no arguments and create an object with nothing in it. When you call encrypt, you pass the message and the key and the method returns the encrypted message.

2:33
In the new way, each CaesarCipher object contains a key. Now when you do new CaesarCipher, you pass in the key and the object you have created stores that key inside of itself. When you call encrypt on such an object you pass in only the message. The key is already in the object and it still returns the same encrypted message as before.

2:56
So if these two implementations produce the same result what are the benefits of an object oriented approach? When you encapsulate the key inside of the cipher object, you have one thing that is capable of taking a message and encrypting it. That makes a nice logical unit to think about, a thing that does a task. You do not need to separately track the key and pass it in. For small programs, such as the ones you have written so far, that may not seem like a big deal. However, as you solve larger programs with more complex code, this design idea will help you immensely. Now that you have seen an example of these ideas in action, the rest of the lesson will teach you about the details of the technique you just saw.

Downloads

Lecture Video mp4

Subtitles (English) WebVTT

Transcript (English) txt

Would you like to help us translate the transcript and subtitles into additional languages?