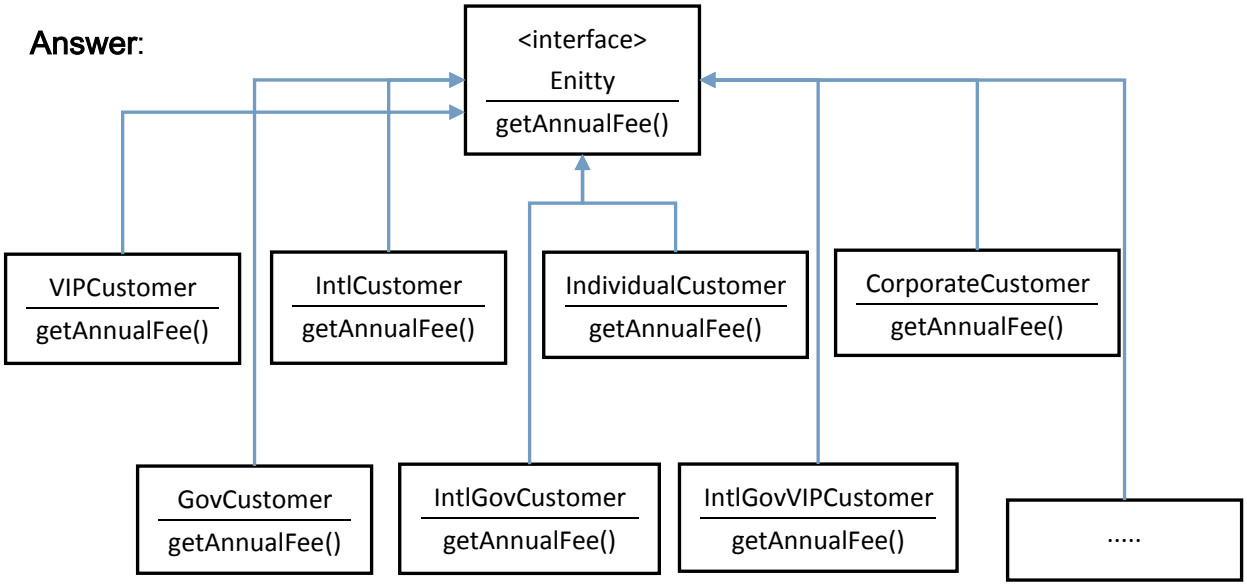
Question1: Show and explain how your code looks like if the Decorator design pattern is not used.

Answer:



Question2: Explain how the Decorator design pattern eliminates conditional statements.

Answer: Using Decorator design pattern, we only need to create the kernel classes and decorator classes. Then we can get any combination instance we want.

For example: if we want to create a international government customer. We can do it as following:

```
Entity customer =new InternationalDecorator(new GovernmentDecorator(new Customer() ) );
```

Question3: Explain why Java API designers decided to use *Decorator* in java.io.

Answer: An instance of InputStream (or outputStream) can be a combination of an InputStream (or outputStream) class and some filters. And in java, there are many inputStream (or outputStream) classes and filters. So the number of the combinations of them is huge. Using Decorator design pattern is the best way to reduce the number of classes in java.io. Developers can use kernel classes(InputStream/outputStream) and filters to create any combination they want.