

In this example, the `open_from_qr_code` method both creates a QR code scanner object and then uses that scanner to obtain the URL to navigate to. This makes the `Browser` class harder to test because now you'd have to patch the `QRScanner` object in order to test `open_from_qr_code`. Also, it's not possible to replace the QR code scanner with another scanner (for example, one that uses another camera than the main front or back camera of the device).

You can solve this by moving the creation of the `QRScanner` out of the `Browser` class and pass it as an argument instead:

```
class Browser:
    def open(self, url: str) -> None:
        print(f"Opening {url} in the browser.")

    def open_from_qr_code(self, qr: QRScanner) -> None:
        qr.choose_camera(Camera.BACK)
        url = qr.scan()
        self.open(url)

def main() -> None:
    print("Navigating to website on device.")
    browser = Browser()
    qr = QRScanner()
    browser.open_from_qr_code(qr)
```

What you see is that in the updated version of the code, creating objects now happens in a single place: the `main` function. This is great, because it gives you as a developer a lot more control over when and where resources are created since all of that is managed in a single place. This is what makes the Separate Creation From Use principle so powerful!