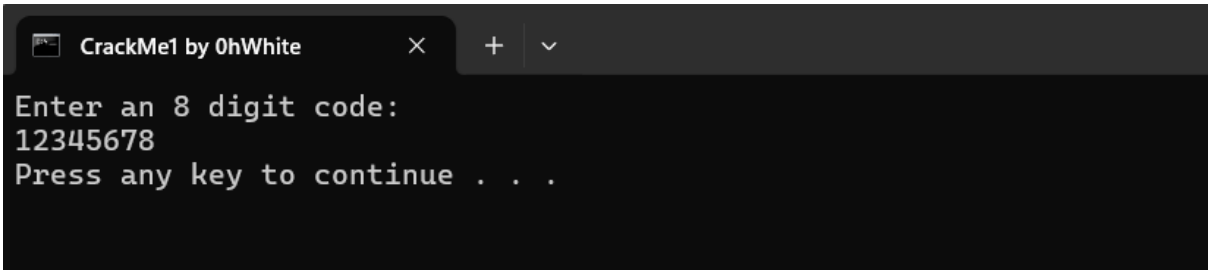
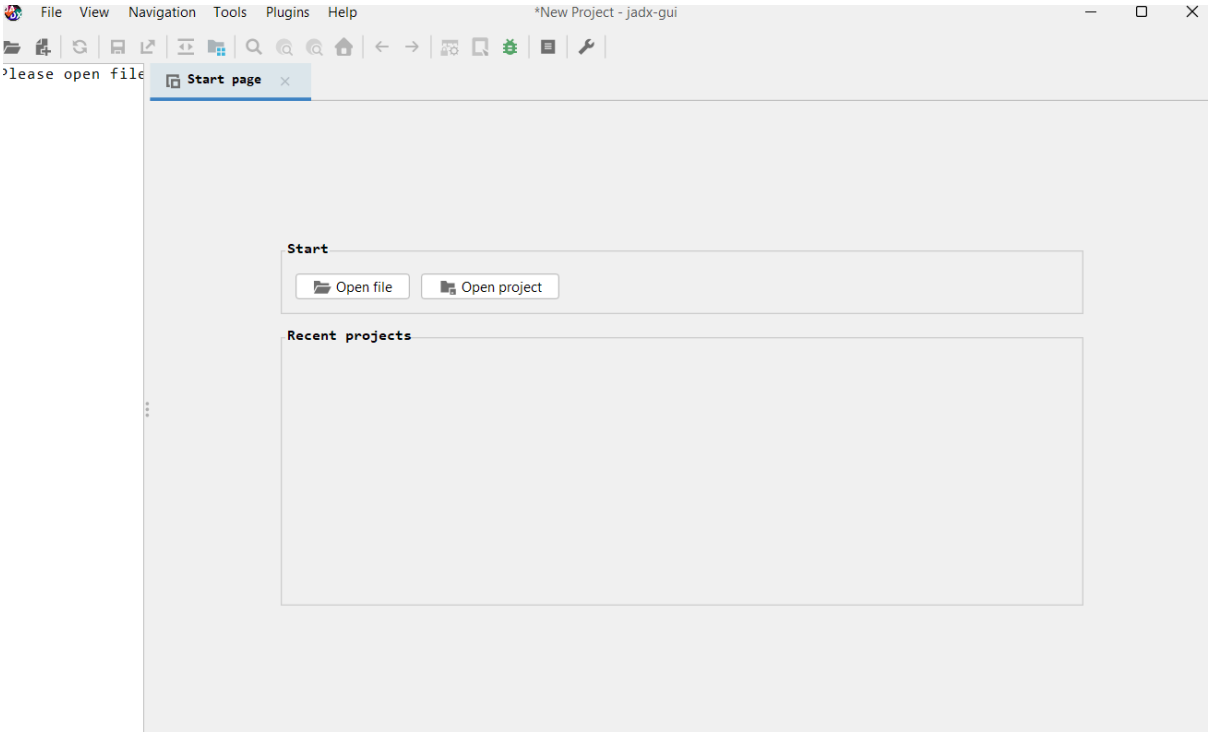
Walking you through your first Java Crackme

In this tutorial, I'll guide you through the process of reverse engineering a Java crackme using JADX. Crackmes are excellent tools for improving software analysis skills, and this particular one is a great starting point. We'll decompile the JAR file, navigate the obfuscated code, and uncover the PIN validation mechanism. Let's dive in and crack this challenge together!

After running the crackme, we are told to enter an 8 digit code. I entered '12345678' just to get a feel for the executable. It appears that when the code is incorrect, the code returns.

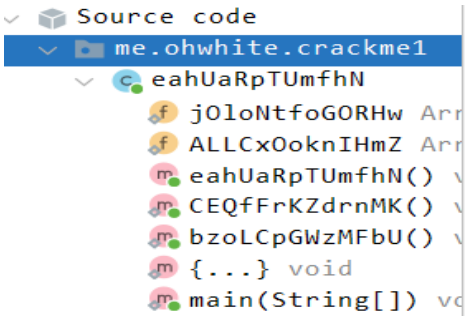


Upon opening JADX GUI, we are going to open the file by dragging the JAR into the box (at least that's how I like to do it).



By looking at the directory on the left, we need to look for anything that may be useful to us. There's a lot to filter through, but as a first Java crackme, I decided to use one which was

quite clear. By looking into the source code path, we can see a folder containing only one class. This is most likely to be where the main class is.



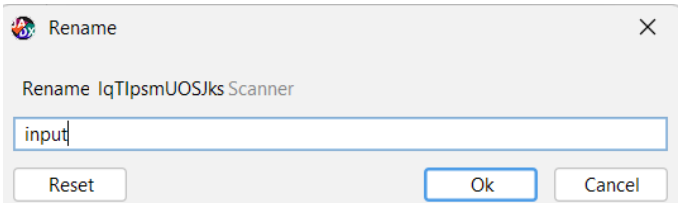
Now we can actually see the Java code. As you would have noticed, most of the method and variable names are obfuscated. Luckily, we can use our deduction skills to find out what they do anyway.

```java
package me.ohwhite.crackme1;

import java.util.ArrayList;
import java.util.Scanner;

/* loaded from: CrackMe1.jar:me/ohwhite/crackme1/eahUaRpTUmfhN.class */
public class eahUaRpTUmfhN {
    static ArrayList<Integer> jOloNtfoGORHw = new ArrayList<>();
    static ArrayList<String> ALLCxOoknIHmZ = new ArrayList<>();

    public static void main(String[] SqbnompFlDpDc) {
        CEQfFrKZdrnMK();
        bzoLCpGWzMFbU();
        System.out.println(ALLCxOoknIHmZ.get(0));
        Scanner lqTIpsmUOSJks = new Scanner(System.in);
        try {
            int hVGPdJleexhgA = lqTIpsmUOSJks.nextInt();
            if (hVGPdJleexhgA != jOloNtfoGORHw.get(0).intValue()) {
                return;
            }
        } catch (Exception e) {
            System.exit(-7);
        }
        System.out.println(ALLCxOoknIHmZ.get(1));
    }

    public static void bzoLCpGWzMFbU() {
        jOloNtfoGORHw.add(5256);
    }

    public static void CEQfFrKZdrnMK() {
        ALLCxOoknIHmZ.add("Enter an 8 digit code: ");
        ALLCxOoknIHmZ.add("You have successfully logged in!");
    }
}
```

The first thing that comes to me is the scanner which is being set as the input of the command line. This is likely to be the user's pin input.

```
        Scanner lqTIpsmUOSJks = new Scanner(System.in);
```

One useful feature of JADX is that we can rename this to be whatever we want, so that we can make it easier to read and analyse. We can do this by clicking on the name we want to change and pressing 'N', alternatively, we can right click and press on rename.



Now, this is just calling 'nextInt' on our Input and assigning it to another variable. We can assume that this is the input that has just been parsed for processing. Let's rename it.

```
int hVGPdJleexhgA = input.nextInt();
```

Now with this section, we can see that the input is being compared to an integer. If they are not equal, then the code returns. We know that this is exactly what happens when we run the code. As well, it appears to be getting the first item out of a list instead of directly comparing the integer. This is another obfuscation technique to throw us off.

```
try {
    int inputParsed = input.nextInt();
    if (inputParsed != jOloNtfoGORHw.get(0).intValue()) {
        return;
    }
```

I renamed the list to something a bit clearer and noticed that these are the only two other instances where this list is mentioned. As we can see, it is declared as an empty list, and then one value is added to the list. Now, the function in which this happens is called before the comparison of the inputs. This confirms that this is the first (and only value in the list)...

```
static ArrayList<Integer> codeArrayList = new ArrayList<>();
```

…

```
public static void bzoLCpGWzMFbU() {
    codeArrayList.add(5256);
}
```

Therefore we can clearly tell that this is likely to be the pin. Let's just test it to be sure.

```
Enter an 8 digit code:
5256
You have successfully logged in!
Press any key to continue . . .
```

You’ve just completed a Java crackme using JADX GUI! I would like to mention that 0hWhite on crackmes.one is the creator of this crackme, so I’d just like to give him a small mention :)