
回到首页

芋道源码 —— 知识星球

我是一段不羁的公告！
记得给芳芳这 3 个项目加油，添加一个 STAR 噢。
https://github.com/YunaiV/SpringBoot-Labs
https://github.com/YunaiV/onemall
https://github.com/YunaiV/ruoyi-vue-pro
2019-10-04
JDK

精尽 JDK 源码解析 —— 调试环境搭建（二）进阶

0. 概述

在 《精尽 JDK 源码解析 —— 调试环境搭建（一）入门》 中，我们已经能够简单的调试。但是，有些跟我一样“杠”的胖友，我要调试 JDK 源码，还要使用自己编译出来的 OpenJDK 。

对，所以本文，我们就来自己编译 OpenJDK13 ，并使用它来调试。

另外，因为每个人编译 OpenJDK 碰到的问题都不同，所以芳芳尽量多贴一些芳芳自己碰到的每个问题时，参考的文章。

还有一个，本文是基于 MacOS 系统。嘿嘿，因为手头暂时没有 Windows 的开发环境。

1. 依赖工具

1.1 编译源码

在 《精尽 JDK 源码解析 —— 调试环境搭建（一）入门》 中，我们已经使用 Git 从 https://github.com/YunaiV/openjdk 克隆了 OpenJDK13 的源码，直接使用它。

1.2 Xcode

编译需要用到 Xcode 工具，一般情况下，我们都已经安装的。如果没有安装的胖友，记得安装下。T T 芳芳之前不小心卸载了，一脸懵逼。

建议的话，打开 https://apps.apple.com/cn/app/xcode/id497799835?mt=12 地址，去 App Store 更新到最新版本。

1.3 引导 JDK

> 芳芳：如果嫌弃哔哔的有点长，可以看本小节最后一端。

编译 OpenJDK 时，需要一个引导 JDK（Boot JDK）。具体原因，芳芳看了下 《Ubuntu 下编译 openjdk8》 文章：

> 安装引导 jdk ，openjdk 的源码编译是需要引导 jdk 的，引导 jdk 需要比要编译的 openjdk 低一个版本，比如我们要编译 openjdk8 ，就需要 jdk7 来做引导 jdk ，可以是 openjdk7 ，也可以是 oraclejdk7 。
>
> 那么为什么需要引导 jdk 呢？我们知道 jdk 版本都是在前一个版本的基础上开发出来的，我们要编译的 openjdk8 在开发的时候，是需要用 jdk7 做环境的，在 jdk7 上做开发测试，是在 jdk7 的基础上开发出来的。

但是芳芳自己在编译 OpenJDK13 本地时，并未安装 JDK12 在本地。在 configure 成功，看了下自动使用了本地的 Oracle JDK13 （java version "13-ea" 2019-09-17 Java(TM) SE Runtime Environment (build 13-ea+33) Java HotSpot(TM) 64-Bit Server VM (build 13-ea+33, mixed mode, sharing) (at /Library/Java/JavaVirtualMachines/jdk-13.jdk/Contents/Home)） 。所以芳芳就搜了下，在 《编译 OPENJDK 小记》 文章：

> 还有一个比较坑的地方。以前编译过虚拟机的同学应该都知道，编译虚拟机需要一个 boot jdk ，即为引导jdk。这个 boot jdk 是如果要编译 1.8 版本的jdk，需要 1.7 版本的jdk。
>
> 但是笔者编译 openjdk1.9 的时候，总是报错 boot jdk 版本不对，非常蛋疼（笔者机器上装的1.8）。一开始我以为是路径不对，改了几次发现，其实编译会自动扫描 JAVA_HOME 下的jdk版本并且找一个合适的，在此提醒各位，如果发现有类似的问题，赶紧按照提示用 brew cask install java 下载一个最新版本的 jdk 就好了。

所以呢，引导 JDK 这步的安装，胖友可以先忽略。在出现问题的时候，可以直接使用 brew cask install java 安装一个最新版本的 JDK ，也是可以引导成功的。

1.4 autoconf

```
brew install autoconf
```

如果不安装 autoconf 的话，待会在 make 时会找不到配置文件。

2. 开始编译

2.1 configure

在 OpenJDK 根目录下，执行命令如下：

```
# 这一句，仅仅是为了告诉胖友，一定要在根目录下
$ pwd
/Users/yunai/Java/openjdk13

# 如下命令，才是正觉
$ bash configure --enable-debug --with-jvm-variants=server --enable-dtrace
```

接着等待一会，如果出现如下日志，说明 configure 成功了：

```
Configuration summary:
* Debug level:    fastdebug
* HS debug level: fastdebug
* JVM variants:   server
* JVM features:   server: 'aot cds cmsgc compiler1 compiler2 dtrace epsilongc g1gc graal jfr jni-check jvmci jvmti ma
* OpenJDK target: OS: macosx, CPU architecture: x86, address length: 64
* Version string: 14-internal+0-adhoc.yunai.openjdk13 (14-internal)

Tools summary:
* Boot JDK:       java version "13-ea" 2019-09-17 Java(TM) SE Runtime Environment (build 13-ea+33) Java HotSpot(TM) 6
* Toolchain:      clang (clang/LLVM from Xcode 11.0)
* C Compiler:     Version 11.0.0 (at /usr/bin/clang)
* C++ Compiler:   Version 11.0.0 (at /usr/bin/clang++)

Build performance summary:
* Cores to use:   12
* Memory limit:   32768 MB
```

不过往往，都是会碰到一些报错。芮芮在如下列举了部分，胖友逐个对照排查。如果没有，可以在星球里留言提问。

报错 1

报错如下：

```
configure: error: No xcodebuild tool and no system framework headers found, use --with-sysroot or --with-sdk-name to
```

参考文章 《如何在 macOS 中编译 OpenJDK10 源代码》 。

输入如下命令：

```
sudo xcode-select --switch /Applications/Xcode.app/Contents/Developer
```

报错 2

报错如下：

```
configure: error: Unable to determine SYSROOT and no headers found in /System/Library/Frameworks. Check Xcode configu
```

参考文章 《build fails with OSX Mojave》 。

输入如下命令：

```
open /Library/Developer/CommandLineTools/Packages/macOS_SDK_headers_for_macOS_10.14.pkg
```

2.2 make

在 OpenJDK 根目录下，执行命令如下：

```
$ export LANG=C
$ make all
```

接着好久一会，如果出现如下日志，说明 make 成功了：

```
Creating jdk image
Creating CDS archive for jdk image
Stopping sjavac server
Finished building target 'all' in configuration 'macosx-x86_64-server-fastdebug'
```

到 build 目录下，我们来看看我们编译的 java 的版本：

```
$ build/macosx-x86_64-server-fastdebug/jdk/
$ bin/java -version

openjdk version "14-internal" 2020-03-17
OpenJDK Runtime Environment (fastdebug build 14-internal+0-adhoc.yunai.openjdk13)
OpenJDK 64-Bit Server VM (fastdebug build 14-internal+0-adhoc.yunai.openjdk13, mixed mode)
```

T T 有一点很尴尬，实际我们使用的是 OpenJDK14 的 EA（Early Access抢 先体验版）。

一般情况下，这步是不太会碰到报错，但是也可能会碰到一些报错。芳芳在如下列举了部分，胖友逐个对照排查。如果没有，可以在星球里留言提问。

报错 1

报错如下：

```
Undefined symbols for architecture x86_64:
  "_objc_loadClassref", referenced from:
      __ARCLite__load() in libarclite_macosx.a(arclite.o)
ld: symbol(s) not found for architecture x86_64
```

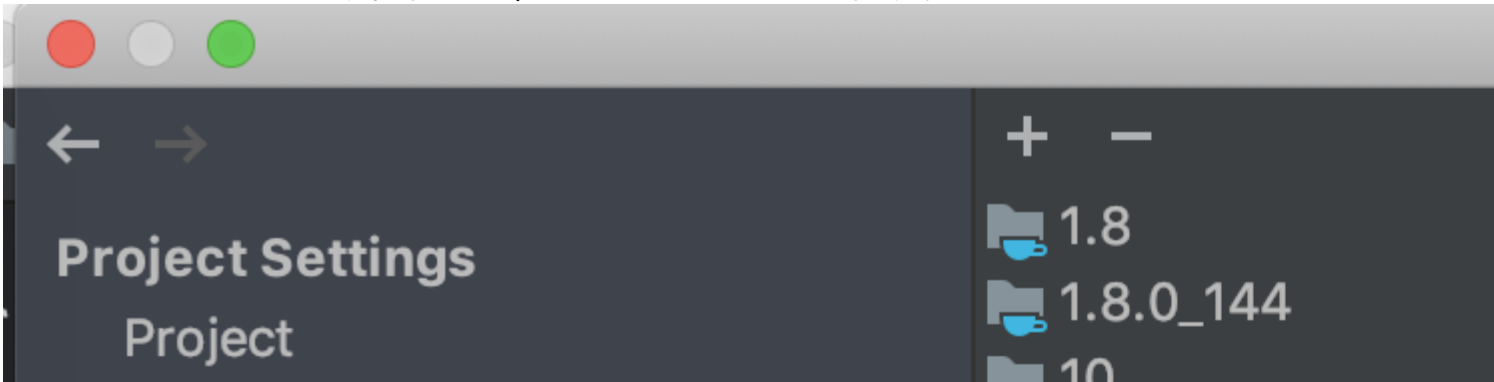
参考文章 https://bugs.openjdk.java.net/browse/JDK-8231572

删除 Lib-java.base.gmk 文件中的 -fobjc-link-runtime 代码。可以看看芳芳 Git 的 461da7026034fdd63df8a9776d0b11895f361523 提交。

3. 使用调试

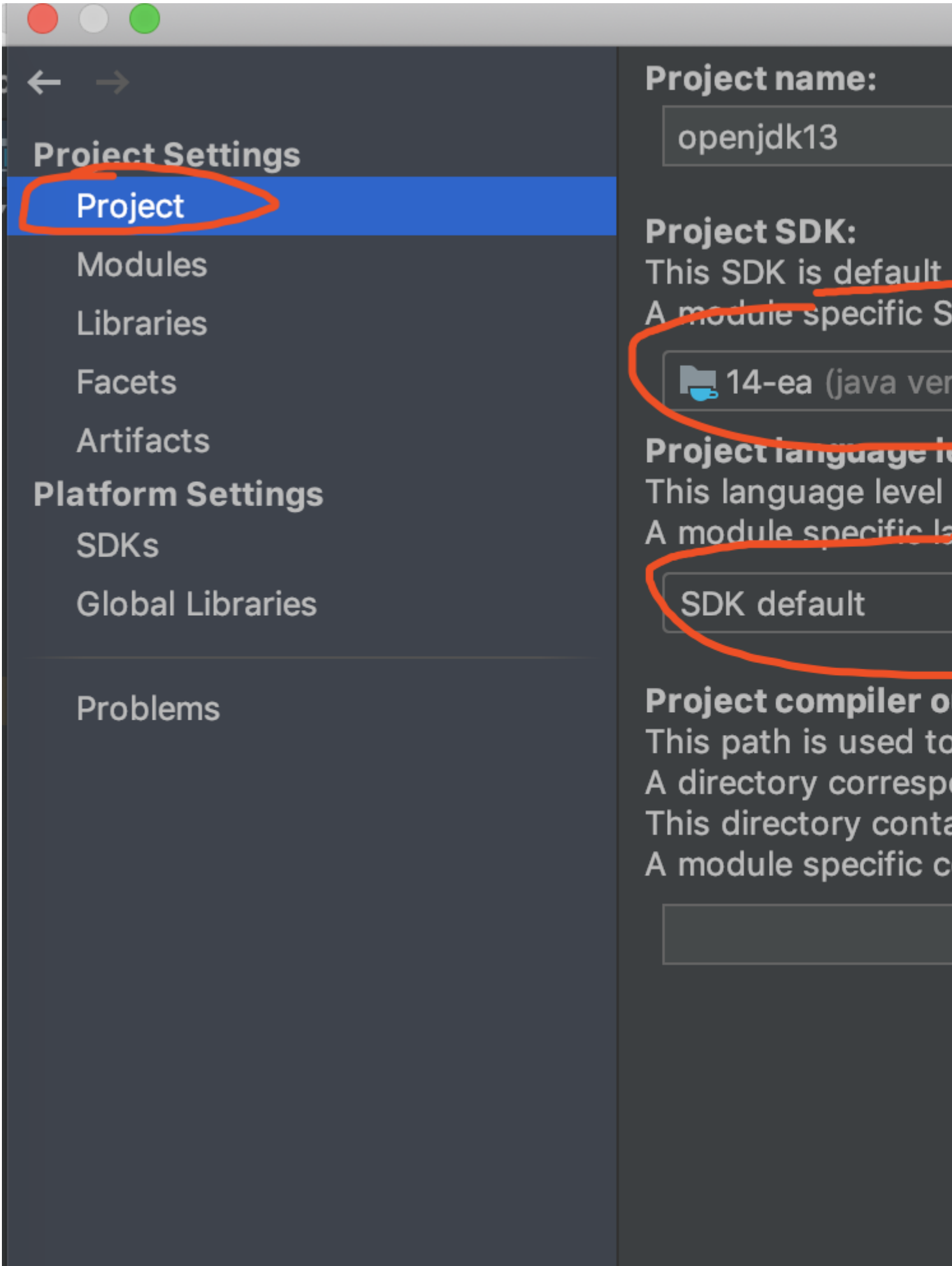
SDKs

在 IDEA 中，创建我们编译出来的 OpenJDK 的 SDKs ，如下图所示：



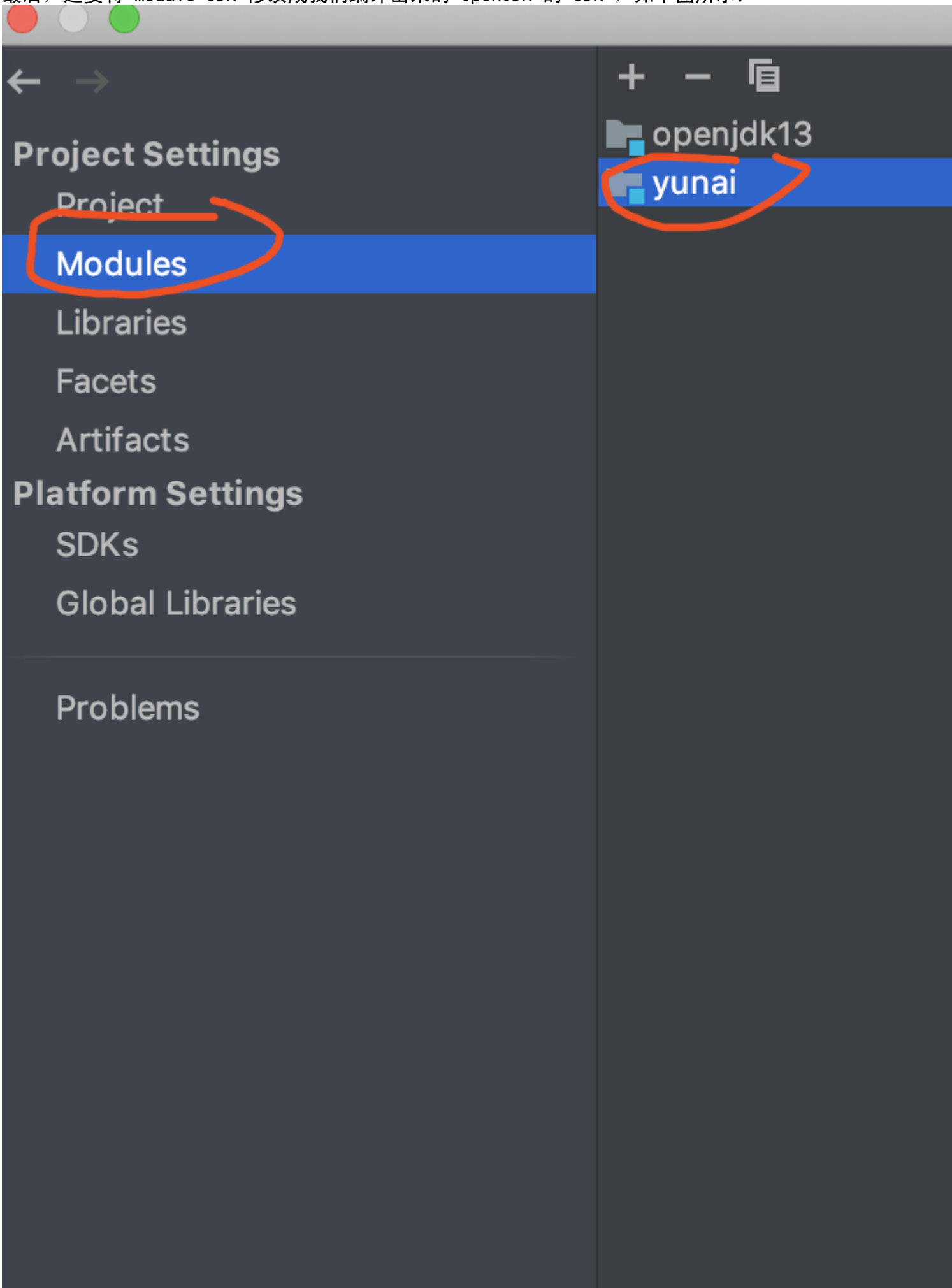
Project SDK

然后，记得将 Project SDK 修改成我们编译出来的 OpenJDK 的 SDK ，如下图所示：



Module SDK

最后，还要将 Module SDK 修改成我们编译出来的 OpenJDK 的 SDK ，如下图所示：



之后，我们就可以愉快的调试了。

666. 彩蛋

编译的过程中，可能每个人碰到的问题不同，如下是艿艿在编译过程中，参考过的文章：

> 《Ubuntu 下编译 openjdk8》
> 《Ubuntu 下编译 openjdk11》
> 《在 MAC 上编译 JDK》
> 《JVM(一):java 技术体系与编译 openjdk》
> 《编译 openJdk10》

虽然 2013 年的时候，在 Ubuntu 上成功编译过一次 OpenJDK 。但是时隔 6 年之后，整个过程还是磕磕碰碰的。

国庆的最后一天假期的 21 点 55 分，准备休息，放空自己。

文章目录