《开源软件设计与开发》课程总结

姓名：毕倪飞　　学号：51195100001

1、开源理解

开源：在互联网领域，可以简单理解为是“开放源代码”的简称。通常来说，指的是将软件项目的源代码向大众开放，允许大众获取、使用、修改和发行。

开源项目：简单理解就是，开放源代码的软件项目。可以认为，开源项目的产出物是软件程序，这包括一个开源项目，可以不断对一款开源软件进行维护和升级，或者有可能在一个开源项目中，产出多款不同的开源软件（但很可能彼此有联系）。

开源软件：直接的字面意思是公开源代码的软件，也就是说，如果软件的源代码是开源的，那么这个软件就可以称之为开源软件。不过，对于很多商业公司来说，开源软件，只能看作是某个开源项目给出的“软件示例”而已，因为软件的源代码已经开放出来了，那么这些商业公司，完全可以根据自己的需要，基于这个示例，修改或衍生出真正适合自己的软件产品。

开源社区：为某个开源项目的开发成员提供的一个学习和交流的空间。由于开源项目常常需要散布在全世界的开发人员共同参与推进，所以开源社区就成了他们沟通交流的必要途径。

开源协议：指开源软件所遵循的许可协议，获得了开源软件的用户，需要在该协议的允许范围内对软件的源代码进行使用、修改和发行。

GitHub，是一个面向软件项目的托管平台，可以用于托管各种类型的软件项目，包括开源项目和私有项目。由于大量开源项目基于 GitHub 进行托管，方便来自世界各地的开发人员共同工作以及获取开源软件，所以在开源项目领域，GitHub 的影响力很大，是开源项目的首选托管平台。

开源软件虽然通常都是免费的，但并不等于软件的开发者们（开源社区）完全放弃了自己的权利和对软件的控制。为了保证开源软件不被一些商业机构或个人窃取，成为他们不劳而获的牟利工具，并影响开源项目的长远发展，开源社区开发出了各种开源协议，用于维护自己的软件版权。在开源协议里面，会详尽表述使用者在获得代码后拥有的权利和义务，包括可以进行何种操作，而何种操作又是被禁止的。开源协议种类非常之多，并且同一款协议会有很多个变种版本。开源协议规定得太宽松，会导致开发者们丧失对开源软件的很多权利，而太严格又不便于使用者们的使用以及开源软件的传播。

常见的开源协议有：GPL、LGPL、BSD、Apache 2.0、MIT

GPL：Linux 就是采用了 GPL 协议。GPL 协议允许代码的获取、代码的免费使用和引用、代码的修改和衍生，但要求对修改和衍生代码的进行开源，不允许修改和衍生的代码做为私有闭源的商业软件发布和销售。这也就是为什么我们能用各种免费的 linux 操作系统，以及 linux 上各种各样的由个人，组织，以及商业软件公司开发的免费软件了。GPL 协议的主要内容是，只要在一个软件中使用到了包含 GPL 协议的产品，则该软件产品必须也采用 GPL 协议，既必须也是开源和免费。由于 GPL 严格要求使用了 GPL 类库的软件产品必须使用 GPL 协议，对于使用 GPL 协议的开源代码，商业软件或者对代码有保密要求的部门就不适合集成/采用作为类库和二次开发的基础。

LGPL：LGPL 是 GPL 的一个为主要为类库使用设计的开源协议。和 GPL 不同，LGPL 允许商业软件通过类库引用方式使用 LGPL 类库而不需要开源商业软件的代码。这使得采用 LGPL 协议的开源代码可以被商业软件作为类库引用并发布和销售。但是如果修改 LGPL 协议的代码或者衍生，则所有修改的代码，涉及修改部分的额外代码和衍生的代码都必须采

用 LGPL 协议。因此 LGPL 协议的开源代码很适合作为第三方类库被商业软件引用，但不适合希望以 LGPL 协议代码为基础，通过修改和衍生的方式做二次开发的商业软件采用。

BSD：BSD 开源协议是一个给予使用者很大自由的协议。开发者可以自由使用和修改源代码，也可以将修改后的源代码作为开源或者专有软件再发布。但是有以下几个要求：如果再发布的产品中含有源代码，则在源代码中必须带有原来代码中的 BSD 协议。如果再发布的只是二进制类库/软件，则需要在类库/软件的文档和版权申明中包含原有代码中的 BSD 协议。不可以用开源代码的作者/机构名字和原来产品的名字做市场推广。BSD 代码鼓励代码共享，但需要尊重代码作者的著作权。BSD 由于允许使用者修改和重新发布代码，也允许使用或在 BSD 代码上开发商业软件发布和销售，因此是对商业集成很友好的协议。而很多的公司企业在选用开源产品的时候都首选 BSD 协议，因为可以完全控制这些第三方的代码，在必要的时候可以修改或者二次开发。

Apache 2.0：Apache Licence 2.0 的简称，Apache Licence 是著名的非盈利开源组织 Apache 采用的协议。该协议和 BSD 类似，同样鼓励代码共享和最终原作者的著作权，同样允许源代码修改和再发布。但是也需要遵循以下条件：需要给代码的用户一份 Apache Licence。如果修改了代码，需要再被修改的文件中说明。在衍生的代码中需要带有原来代码中的协议，商标，专利声明和其他原来作者规定需要包含的说明。如果再发布的产品中包含一个 Notice 文件，则在 Notice 文件中需要带有 Apache Licence。你可以再 Notice 中增加自己的许可，但是不可以表现为对 Apache Licence 构成更改。Apache Licence 也是对商业应用友好的许可，使用者也可以在需要的时候修改代码来满足并作为开源或商业产品发布/销售。

MIT：MIT 是和 BSD 一样宽泛的许可协议,源自麻省理工学院。使用 MIT 协议的开源软件作者只保留版权,而对使用者无任何其它限制。MIT 与 BSD 类似，但是比 BSD 协议更加宽松，是目前最少限制的协议。这个协议唯一的条件就是在修改后的代码或者发行包中包含原作者的许可信息，且适用于商业软件。

2、开源贡献

本学期我参与的开源项目是腾讯的一套高性能 RPC 框架 Tars，我从 github 上找到了 Tars 并克隆到了本地，根据官方文档进行了安装部署，并尝试了一个快速入门的例子。但是在代码贡献方面没有做出贡献，只是加入了一个 Tars 的交流群，在群里讨论了几个关于 Tars 的问题。

```
public static void main(String[] args) {
        CommunicatorConfig cfg = new CommunicatorConfig();
        //构建通信器
        Communicator communicator =
CommunicatorFactory.getInstance().getCommunicator(cfg);
        //通过通信器，生成代理对象
        HelloPrx proxy = communicator.stringToProxy(HelloPrx.class,
"TestApp.HelloServer.HelloObj");
        proxy.async_hello(new HelloPrxCallback() {

                @Override
                public void callback_expired() {
                }

                @Override
                public void callback_exception(Throwable ex) {
                }

                @Override
                public void callback_hello(String ret) {
                        System.out.println(ret);
                }
        }, 1000, "HelloWorld");
}
```

3、课程反馈

老师讲课非常生动，能够激发学生的兴趣。

课堂上老师还请到了各路大神为我们带来了各种精彩的演讲与讨论，开阔了我们的眼界，这一点非常好。

老师上课的形式丰富多样，课堂上及时交流，课堂下有调查问卷形成了充分的反馈。

4、参考文献

【1】Tars 文档：https://tarscloud.gitbook.io/tarsdocs/