

Методы и средства программной инженерии

Лабораторная работа №3

Вариант №2332

Преподаватели: Покид Александр Владимирович

Выполнили: Груздева Евгения Николаевна,

Липонина Екатерина Алексеевна

N3248

Санкт-Петербург

2022

Написать сценарий для утилиты [Apache Ant](http://ant.apache.org), реализующий компиляцию, тестирование и упаковку в jar-архив кода проекта.

Каждый этап должен быть выделен в отдельный блок сценария; все переменные и константы, используемые в сценарии, должны быть вынесены в отдельный файл параметров; MANIFEST.MF должен содержать информацию о версии и о запускаемом классе.

**Cценарий должен реализовывать следующие цели (targets):**

1. **compile** -- компиляция исходных кодов проекта.
2. **build** -- компиляция исходных кодов проекта и их упаковка в исполняемый jar-архив. Компиляцию исходных кодов реализовать посредством вызова цели **compile**.
3. **clean** -- удаление скомпилированных классов проекта и всех временных файлов (если они есть).
4. **test** -- запуск junit-тестов проекта. Перед запуском тестов необходимо осуществить сборку проекта (цель **build**).
5. **doc** - добавление в MANIFEST.MF MD5 и SHA-1 файлов проекта, а также генерация и добавление в архив javadoc по всем классам проекта.
6. **report** - в случае успешного прохождения тестов сохраняет отчет junit в формате xml, добавляет его в репозиторий svn и выполняет commit.

<https://habr.com/ru/post/323204/>

<https://coderlessons.com/tutorials/java-tekhnologii/izuchite-apache-ant/kratkoe-rukovodstvo-po-ant>

JAR расшифровывается как Java ARchive — архив Java. Если не особо вдаваться в подробности , то JAR-файл представляет собой обычный ZIP-файл с некоторыми дополнениями. Основная задача этого архива — хранить файлы с классами. Т.е. пакеты/каталоги, внутри которых находятся class-файлы архивируются и JVM может их использовать уже в более удобном (компактном) виде.

<https://habr.com/ru/post/169381/>

**Модульное тестирование**, или **юнит-тестирование** (англ. unit testing) — процесс в программировании, позволяющий проверить на корректность отдельные модули исходного кода программы.

Идея состоит в том, чтобы писать тесты для каждой нетривиальной функции или метода. Это позволяет достаточно быстро проверить, не привело ли очередное изменение кода к регрессии, то есть к появлению ошибок в уже оттестированных местах программы, а также облегчает обнаружение и устранение таких ошибок.

<https://ant.apache.org/manual/Tasks/junit.html>

<https://github.com/junit-team/junit4/wiki/getting-started>

<https://java-online.ru/java-javadoc.xhtml#javadoc_ant>

<https://java-online.ru/blog-ant.xhtml>