**Practice 1 (JSON & XML Serialization/deserialization):**

1. Create/parse a JSON file (<https://www.sitepoint.com/10-example-json-files/> )
2. **Create a new Datatype in Java (Student, Car, Book etc) with at least 5 properties, 2 of them are non-primitives(another datatype) and at least one - array. Serialize and deserialize objects of this type in/from XML and JSON (2 ways for each)**

**Practice 2 (Rest WS with XML):**

* **Read a set of dates from file (txt/csv/xls)**
* **Use these dates as parameters to request currency rates from BNM site (<https://bnm.md/en/official_exchange_rates?get_xml=1&date=13.02.2018>)**
* **Create a list of objects based on received values**
* **Save these values on separate page (sheet) for each date in xls/xlsx file**

**Practice 3 (Rest WS with JSON)**

* **<https://http.cat/> (save to file)**
* **<https://api.coindesk.com/v1/bpi/currentprice.json>**
* **<https://yesno.wtf/api> (save each answer in separate folder with file - 1000 times)**

**Practice 4:**

* Test case to complete form
  + <http://demoqa.com/registration/>
* <http://adam.goucher.ca/parkcalc/index.php>
  + Test with TestNG DataProvider

**Practice 5:**

1. Create a site (wordpress.com or other) – manual

* practice5975143571.wordpress.com: practice5975143571/asdfqwer1234

1. Use existing solution (uploaded to drive)
2. Use library of custom controls (wrappers for TypifiedElement from htmlelements - WebTextInput, WebLink, WebButton, WebTextBlock, WebCheckBox, WebRadio) - each control contains needed log messages and asserts
3. BaseTest and Abstract POM, Helpers
4. log4j2 - configure to use 2 appenders: console and log file (each time new log file with date/time name)
   1. Optional: log name using test name and date/time
5. Use config/config xml/json file for framework
   1. url
   2. username
   3. password
6. Organize logs and screens in a separate folder/folders
7. Populate POMs with needed elements
8. Select browser from config
9. Automate 8 scenarios

Scenarios:

* Check login / sign out
* Create and publish new Post, check presence in table
* Create and publish new Page, check presence in table
* Create and move to draft Post, check presence in table
* Create, preview and publish new Page, check presence in table
* Invite new User (check presence of options in dropdown)
* Edit existing post (many runs with different data - timestamp/random), validate updated post using View Button - validate before and after
* Create and delete page

**Practice1:**

1. jsonParserCreator: MyJSONCreator, MyJSONParser, MyJSONQuery - название говорит само за себя

2. serializationDeserialization: тип Student содержащий массив оценок и 2 других типа данных: Address и Parents. Json\_Gson, Json\_Jackson, Xml\_Jackson, Xml\_Xtream сериализация / десериализация в/из json/xml используя соответствующую библиотеку - Gson, Jackson, Xtream

**Practice2:**

Исходные даты находятся в dates.txt. Запускать надо Practice2DriverClass. Результат записывается в CurrencyRates.xlsx.

**Practice 3 (Rest WS with JSON)**

* Cats - записывает картинки 5 случайно выбранных кода в папку Cats
* CoindeskDriverClass - так как подробного задания не было, я просто послала запрос, вывела на экран полученный json, десериализовала его и вывела полученный обьект
* YesnoDriverClass - сохраняет в папку Yesno 5 картинок с соответствующими json-ами (каждую пару отдельно). У меня была проблема с handshake-oм. Если будет и у тебя надо будет скопировать определенные файлы в нужное место. Вот решение: https://stackoverflow.com/questions/46081424/java-8-recv-tlsv1-2-alert-fatal-handshake-failure

**Practice 4:**

FormFillingTest - заполняет анкету

ParkingCostTest - тестит 5 вариантов парковки

**Practice 5**

Попыталась создать свой сайт, но там вроде как совсем другая структура, так что взяла твой (все-таки задание не на создание сайта :-)). Немного переделала структуру фреймворка, в частности избавилась от папки core: переместив существующие там классы по другим папкам (мне так удобнее), а методы из Helpers записала в WebTypifiedElement, так как они только там и используются. Заодно избавилась от static-а который мешал мне при логах.

Если будешь гонять на Mac-е, надо будет перекомментить пути к веб-драйверам (это в BaseTest.setupEnvironment) и возможно поменять слеши в путях (это там же и в начале BaseTest-а, где указываются все пути). Я пыталась писать наименее ОС-зависимо, но не знаю насколько получилось...

Запускать надо testng.xml. В результате считываются конфиги из config.json и ранятся все тесты, записывая логи для каждого отдельно в папку logs, а в случае фэйла, делается скриншот и сохраняется в папку screenshots. Перед каждым запуском testng.xml папки logs и screenshots очищаются.

Тесты разумеется не исчерпывающие :-)