### 00\_intro.md Intro

While going tru this training you will develop console online store. Each next task will append some functionality or flexibility to it;

Each task has a description and materials, it doesn't mean that you should read this or only this materials,  you are free to 'google' topics and select the most preferable content.

Process

A trainee must read and understand all materials related to the task and then complete all requirements from the task file.

A link to the pull request must be sent to the trainer via Email or Skype. The trainer performs the code review, a trainee must update code according to the comments from the code review and then inform his/her trainer that additional code review is required in the scope of the same pull request.

Administrative part

A trainee is obligated to inform his trainer about vacations or other activities, which will stop training process for 1 week and more. If there is no obstacles, then it is advised to complete at least one task per week (depends on the task scope). If a trainee is assigned on a project, then he/she should work on the training in his/her own time. Training must be completed in < 3 months. If training takes > 3 months, then situation will be reviewed individually (continue or stop the training and find the cause of a slow progress). If a trainee has difficulties, then he/she can ask the trainer for advice or a hint (only about the task itself). The final task is the end of the training. Basing on this task the trainer forms feedback for stakeholders, therefore a trainee works on it without any help.

Before start

**Knowledge**

Before starting this course you should be familiar with:

* Programming in general
* How to write and launch java application
* Java classes, methods, variables, exceptions
* Java if/else, switch, loops

All this topics you can discover before the course [here](https://www.homeandlearn.co.uk/java/java.html) (Sections 1-8)

**Software**

Before start you should install:

1. [Java 1.8](https://www.oracle.com/java/technologies/javase-jdk8-downloads.html)
2. [Git](https://git-scm.com/)
3. [IntellijIdea](https://www.jetbrains.com/idea/) Community edition

Good luck!

**01\_maven.md**

Maven

**Materials**

[Maven in 5 Minutes](https://maven.apache.org/guides/getting-started/maven-in-five-minutes.html)

[Maven Getting Started Guide](https://maven.apache.org/guides/getting-started/index.html)

[Naming Maven](http://maven.apache.org/guides/mini/guide-naming-conventions.html)

[Naming Java](https://www.oracle.com/java/technologies/javase/codeconventions-namingconventions.html)

**Task #1**

Before start implementation our OnlineStore, we need to prepare project structure and set up dependency manager. To handle our project dependencies and source code build we will use Maven.

Please crete multi-module maven project in Idea, with such modules:

1. parent (this is general store module)
2. domain
3. store
4. consoleApp

**02\_git.md**

#### Materials

[Git](https://git-scm.com/doc)

[GitHub Hello World](https://guides.github.com/activities/hello-world/)

#### Task #2

We will store source code of our OnlineStore in GitHub. Before start you should read and understand git principals and main git commands.

Please create GitHub repo and append your trainer to this repo!

While doing each task, you should create a separate branch with the name of the topic, e.g. 02\_git, push you task code to this branch and create a pull request from your branch to master branch, and assign it to your trainer.

**03\_oop.md**

OOP

**Materials**

[OOP](https://docs.oracle.com/javase/tutorial/java/concepts/index.html)

[Lecture 3](https://coherentsolutions.sharepoint.com/sites/training-center/_layouts/15/WopiFrame.aspx?sourcedoc=%7b21357CB9-7D9D-4E18-AD42-22ADC9979308%7d&file=L3.pptx&action=default)

[Lecture 4](https://coherentsolutions.sharepoint.com/sites/training-center/_layouts/15/WopiFrame.aspx?sourcedoc=%7b87729213-AD13-40A5-876C-67E647EC725A%7d&file=L4.pptx&action=default)

[Reflection](https://docs.oracle.com/javase/tutorial/reflect/)

[Reflections Lib](https://github.com/ronmamo/reflections)

[Faker](https://github.com/DiUS/java-faker)

**Task #3**

Before start creating source code, read carefully all materials about OOP. It is not only 3 principles for interview;)

Store functionality should be based on above principles.

Classes to create:

* Product with such attributes as [name, rate, price]
* Category classes with the name attribute, for each store category [bike, phone, milk] and products list
* Store - class that should handle category list
* RandomStorePopulator - utility class that will populate out store/category with fake data using Faker lib
* StoreApp - class with main method to execute our store scenario.

When invoke main method, application should init store with categories and products and pretty print this data. Also, categories should be read dynamically (at runtime), from base category package using reflections lib.

**04\_collections.md**

Collections

**Materials**

[Collections](https://docs.oracle.com/javase/tutorial/collections/index.html)

[Lecture 9](https://coherentsolutions.sharepoint.com/sites/training-center/_layouts/15/WopiFrame.aspx?sourcedoc=%7bEF21525C-52B3-45A0-8E14-71BFC9BAB74E%7d&file=L9.pptx&action=default)

[XML](https://en.wikipedia.org/wiki/XML)

[XML processing](https://docs.oracle.com/javase/tutorial/jaxp/)

**Task #4**

Starting extend our store. Please append ability user to interact with our store, while sending commands thru read stream.

Add support of such commands:

* sort - products from store according config. In resources folder create xml config file like
* <sort>
* <name>asc</name>
* <price>asc</price>
* <rate>desc</rate>
* </sort>

Config file can contains from 1 to N(in our case 3 is max) fields. Sort should be done using Comparator. Sort and print should not modify default store product lists and their order.

* top - print top 5 products sorted via price desc
* print - prints all products and categories from previous task
* quit - exit app

**05\_patterns.md**

Patterns

**Materials**

[Patterns](https://refactoring.guru/design-patterns)

**Task #5**

Read all materials, try to find a proper place to your newly learned patterns in our app.

**06\_multithreading.md**

Multithreading

**Materials**

[Concurrency](https://docs.oracle.com/javase/tutorial/essential/concurrency/) [L10 from slide 24](https://coherentsolutions.sharepoint.com/sites/training-center/_layouts/15/WopiFrame.aspx?sourcedoc=%7b64853C24-C830-4C50-B8B4-723AFC490668%7d&file=L10.pptx&action=default) [L11 from](https://coherentsolutions.sharepoint.com/sites/training-center/_layouts/15/WopiFrame.aspx?sourcedoc=%7b0D5F5DD0-CBDE-4EB7-8D17-CDEC874B3F64%7d&file=L11.pptx&action=default)

**Task**

Please implement create order functionality. Each order should be processed in separate thread. When user select product , generate the random int from 1 to 30, and create thread that will process selected order for selected time, and after it place the product in another collection (for example, purchased goods). And create one more thread, that will be executed periodically, e.g. ones in 2 mins, that will clean up purchased collection.

You can implement this in "native" java methods but better and simplier to use [java.util.concurrent](https://habr.com/ru/company/luxoft/blog/157273/) package.