# Item & Category Coding Challenge

# Detail Design Document

Contents

[Introduction 2](#_Toc20168746)

[Problem Statement 2](#_Toc20168747)

[Design 3](#_Toc20168764)

[Implementation Details 4](#_Toc20168767)

[Infrastructure, development tools and Config 5](#_Toc20168769)

[Deployment/Import Details 6](#_Toc20168770)

# Introduction

This represents the use cases covered to implement the item & category coding challenge which is based on the item and category and optimization of items based on item ratings. It also prints the coordinates of picked optimized items.

# Problem Statement

# Imagine you have a basket. Your objective is to fill this basket with as many items as you can. However, there are 20 categories of items and you may only pick 1 item from each category. Each item has a price, a shipping cost and a rating.

# Write a program that adheres to the above constraints and does the following each time you run it.

# ▪ Generates;

# ▪ 20 item categories - (Category1, Category2, .... , Category20)

# ▪ 10 items of each category – (Item 1, Item 2, ...., Item 10).

# ▪ Each item should be randomly assigned;

# ▪ A price of between $1 to $20

# ▪ A shipping cost of between $2 to $5

# ▪ A rating of between 1 to 5 (a bigger value indicates a better rating)

# ▪ Picks as many items as you can for the basket, while keeping the total cost (price + shipping cost)

# of all picked items below $50, and ensuring that the sum of ratings of all items picked is optimized.

# At the end, your program should;

# • Print the coordinates of the selected items.

# e.g. – CategoryA:ItemX,CategoryB:ItemY, ....

# • Total cost, and

# • Sum of ratings of all the items that were picked.

# Design

# Based on the problem statement, the below sequence diagram is created, which basically shows the classes & operations which gets invoked when the Junit is executed.

# 

# Implementation Details

# This represents the source code & operations written to solve the problem statement. The below are the high-level description of resources used to implement the solution of the problem statement.

|  |  |
| --- | --- |
| **File Name** | **Description** |
| ItemCategoryService.java | This is an interface which has all the necessary methods to implement the solution of the problem statement. Below are the methods –  generateItemCategories()  generateItemsOfEachCategory(String categoryId)  pickOptimizedItems() |
| ItemCategoryServiceImpl.java | This class implements the methods declared in the above-mentioned interface along with some private helper methods. |
| ItemCategoryUtil.java | This represents the util class which has the utility methods to get the random numbers. |
| Category.java | This represents the model for the Category. |
| Item.java | This represents the model for the Item. |
| TestItemCategoryService.java | This class instantiates the Impl class and unit test's various methods available to satisfy the business requirement. |
| EntityLogger.java | This logger is created to support logging wherein separate log files can be generated for the same logger category for separate entities. |
| ResourceLoader.java | This class holds the logic to load a resource which have necessary config properties during run time. This is a singleton class which loads only once during the application life-cycle. |
| ItemCategoryConstants.java | This class holds constants that are accessed by various components in the application. |
| pom.xml | This file is located in the root directory of the project. The pom.xml has declaration about the project and various configurations. The pom.xml file also has the plugins. This is responsible for adding jar dependencies and part of Maven. |
| application.properties | This is a resource file which contains the required attributes for the application. |
| testitemcategory.properties | This is a resource file which contains the required attributes for the Junit. |

# [Infrastructure, development tools and Config](file:///\\\\aur.national.com.au\\pages\\createpage.action%3fspaceKey=CBDCDEV&title=TemplateDetailDesign-_Toc514333927Infra&linkCreation=true&fromPageId=74728929)

The solution has been implemented using java 1.8 version. It also has the various dependencies included in the build config file pom.xml. Below are the list –

slf4j version 1.6.1

java. version1.8

Junit version 4.13-beta-1

maven-javadoc-plugin version 3.1.1

# Deployment/Import Details

# Once the project is imported in any IDE the following steps needs to be done –

# Right click on the project and select the option “Update Project” under Maven option.