Technical Design Document

of Container Capacity Calculator

Version : 1.0

Authors : Diganta Mohapatra

Last Reviewed By :

Creation date : 25th Sep 2018

Document Control

Sorted by Cost

| Date | Name | Version | Change Reference |
| --- | --- | --- | --- |
|  |  |  |  |
| 30th Jan 2018 | Diganta Mohapatra | 1.0 | Created Draft Version |
|  |  |  |  |

Reviewers

| Date | Name | Review type | Version | Position |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Table of Contents

Table of Contents

[1 INTRODUCTION 4](#_Toc525652391)

[2 IMPLEMENTATION 5](#_Toc525652392)

[2.1 public class TestMain 5](#_Toc525652393)

[2.2 public class Packer 5](#_Toc525652394)

[2.3 Public class ContainerDetails 7](#_Toc525652395)

[2.4 Public class Container 8](#_Toc525652396)

[2.5 Public class Packet 8](#_Toc525652397)

[3 Deployment and Execution Details 9](#_Toc525652398)

[3.1 Prerequisites 9](#_Toc525652399)

[3.2 Deployment 9](#_Toc525652400)

[3.3 Execution 9](#_Toc525652401)

[4 Appendix 9](#_Toc525652402)

[4.1 Server details 9](#_Toc525652403)

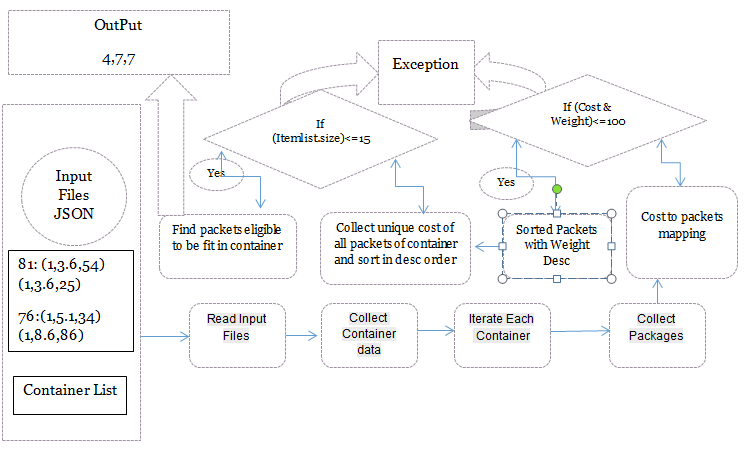
# INTRODUCTION

The purpose of the document is to get the containers and try to fit sorted package items inside the container according to the capacity of the container.

So Basically following steps captured as part of this utility developed.

* Read Input File
* Collect containers
* Iterate each container
* Collect Packets
* Create cost to packets mapping
* Collect unique cost of all packets of container and sort in desc order

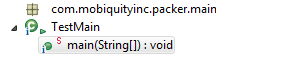
**Dataflow:**



# IMPLEMENTATION

## public class TestMain

**Description :** TestMain is the main class which triggers the module.

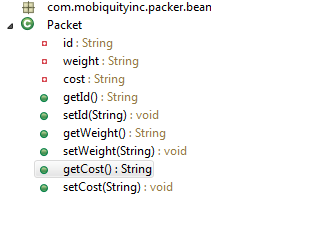


**Operation :** public static void main(String[] args)

**Description :** This is the main method that starts the module. This static method calls the another class static method.

## public class Packer

**Description** This class is used to accept container in json format contains container capacity and package list details like package ID ,weight and cost. Output of this class is to return set of ids wrt to packages which are being sorted according to the weight and cost .Condition is like those packages will be fit which have more cost and weight should be equal or less than capacity of conainer. If multiple items with same cost , ideally program will pick those items those have less weight.



**Operation :** **public** **static** String pack(String fileName)

**Description :** This method accepts input json files as argument and return the sorted output container with set of Index ids.

**Operation :** **private** **static** ContainersDetails readInputFile(String fileName)

**Description :** This method basically read the JOSN and return a Gson object.

**Operation :** **private** **static** Map<String, List<Packet>> getCostToItemMappings(List<Packet> items)

**Description :** This method is used to accept list of items and set the cost as the map key and value will be the list . This method basically return a linked hash map which maintains the insertion order key along with sorted list in descending order.

**Validation:**

if cost and item capacity is more than 100 it will be excluded from the container.

If valid file is not present , then it will throw a custom APIException .

**Operation :** **private** **static** **void** sortedPackagesOnWeights(List<Packet> items)

**Description :** Sorted list of items on basis of input weights.

**Operation :** **private** **static** List<Double> sortedPackageCostInDescOrder(Map<String, List<Packet>> maps)

**Description :** This method will return a list with cost in descending order.

**Operation :** **private** **static** List<Packet> findElementsToBeFilledInContainer(Map<String, List<Packet>> maps, **double** capacity,List<Double> sortedCosts)

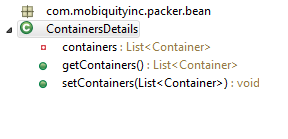
**Description :** This method will accept sorted map, capacity of the container and sorted cost in descending order. Iterate over the sortedcost list until the packages weight reached equal or less than container capacity and return the packet list. A container can pick upto maximum of 15 items and post it will discard rest of the packages although capacity to fit more items inside container.

**Operation :** **private** **static** String itemsToBePacked(List<Packet> items)

**Description :** This method will accept sorted map, capacity of the container and sorted cost in descending order. Iterate over the sortedcost list until the packages weight reached equal or less than container capacity and return the packet list.

## Public class ContainerDetails

**Description:** This method is used to get the container details

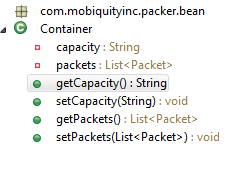


**Operation :** **public** List<Container> getContainers()

**Description :** This method returns list of containers.

## Public class Container

**Description:**This class is responsible to get list of packages and capacity



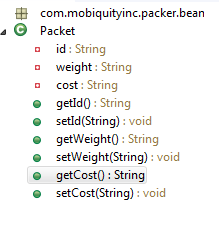
**Operation :** **public** List<Packet> getPackets()

**Description :** This method is used to get list of packets.

**Operation :** **public** String getCapacity()

**Description :** This method is used to get capacity of individual container.

## Public class Packet



**public** String getId(),**public** String getWeight(),**public** String getCost()

**Description:** These methods get packet id ,weight and cost respectively.

# Deployment and Execution Details

This section covers details required to set up and execute the code in any environment.

## Prerequisites

1. Make a main class with will accept static method for execution.
2. Provide a Json file with standard definition and inputs.

## Deployment

1. Extract the JAR.
2. Deploy the stand alone code in specific location where JVM can be accessible so the piece of code will execute.
3. Define a main method and try to provide absolute path for the JSON.

Note: Currently we have setup the test main method and provided a relative JSON path with valid data . (May be you can use for testing and evaluation purpose).

## Execution

* Execute the Jar in any located JVM host.

# Appendix

## Server details

|  |  |  |  |
| --- | --- | --- | --- |
| Environment | Application | GIT Location | User |
|  | Container Calculator |  | mobiquity |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |