Seneca-test

- The problem statement says accept vendors list and their respective percentages, and distribute work loads based on their percentages.
- Here the tricky part is we may end up distributing non uniquely for example you have 10 workloads distributed across 3 vendors one with 35, 45 and 20. 10 work loads will not be defined equally.
- So I followed 2 approaches, I call it as priority and non priority.
- Application is developed in spring boot which runs on 8080 port. I have used lombok for auto generation of getters and setters. If you want to run application in your local, you have to configure lombok in eclipse. https://howtodoinjava.com/automation/lombok-eclipse-installation-examples/
- I haven maven as build tool.

First approach

- In this approach we will have request object as list which has vendor information details sunch as (vendor name and their percentage) which are mandatory. In addition to this we do have total workloads and consider priority which are optional. So we will ignore priority and will do normal manner based on decimal values, ie., 0.1000 will take precedence over 0.3000.
- And if any difference in final list, the values have been adjusted

based on precedence.

- Here is the request object.

```
@Getter
@Setter
@NoArgsConstructor
public class WSVendorDistributeRequest {
    private List<WSVendorRequest> distributions = new Array
List<WSVendorRequest>();
    private Integer totalWorkload;
    private Boolean considerPriority = false;
}
@Getter
@Setter
@NoArgsConstructor
public class WSVendorRequest {
    private String name;
    private Integer percentage;
    private Integer priority;
}
```

Consider json request:

You could see 45%, 10%, 15%, 30% as distributions for vendors.

```
{
    "distributions" : [{
        "name" : "Vendor 1",
```

```
            "percentage" : 45
        },{
            "name" : "Vendor 2",
            "percentage" : 10
        },
        {
            "name" : "Vendor 3",
            "percentage" : 15
        },
        {
            "name" : "Vendor 4",
            "percentage" : 30
        }],
        "totalWorkload" : 77
}
```


Distribtions for above request to vendors. This is without priority.

The allocations happend based on the order of decimal value.

Vendor	Percentage	Distributions
Vendor 1	45%	34
Vendor 2	10%	8
Vendor 3	15%	12
Vendor 4	30%	23

Second Approach

- In this approach, we will give prirority, even though we didn't mention the priority for all objects, it will create priority which are already mentioned and will assign priority for others.
- In request, there is a property by name considerPriority, By default it is false, we have to pass true of you want to use.
- If priority is mentioned, vendor with least priority is adjusted.

Consider above request with priority: You will priority for each, even though if you don't mention priority for each vendor still if "considerPriority" is mentioned, the priority will be assigned based on order in collection.

```
{
    "distributions" : [{
        "name" : "Vendor 1",
        "percentage" : 45,
        "priority" : 2
    },{
        "name" : "Vendor 2",
        "percentage" : 10,
        "priority" : 1
    },
    {
        "name" : "Vendor 3",
```

```
        "percentage" : 15,
        "priority" : 3
    },
    {
        "name" : "Vendor 4",
        "percentage" : 30,
        "priority" : 4
    }],
    "totalWorkload" : 77,
    "considerPriority" : true
}
```


Distribtions for above request to vendors. This is with priority.

The allocations happend based on the order of decimal value.

Vendor	Percentage	Distributions
Vendor 1	45%	35
Vendor 2	10%	8
Vendor 3	15%	12
Vendor 4	30%	22

- Now if you observe response as 30% has least priority, the distribution is adjusted here.

Rest Api documentation

Base Url	Api Path	Method
http://localhost:8080	/api/distribute	POST

Response is :


```
{
    "distributions": [
        {
            "name": "Vendor 5",
            "percentage": 30,
            "distribution": 22
        },
        {
            "name": "Vendor 3",
            "percentage": 15,
            "distribution": 12
        },
        {
            "name": "Vendor 1",
            "percentage": 45,
            "distribution": 35
        },
        {
            "name": "Vendor 2",
            "percentage": 10,
```

```
            "distribution": 8
        }
    ]
}
```

Error Messages :

Reason	Field	Message
If totalWorkload is not set.	totalWorkload	provide total workloads.
If vendor distributions list is empty or null	distributions	provide vendor distributions.
If percentage is greater than or less than 100%	totalPercentage	Percentage sum of all vendors should be 100. But actual is =>${param}

Test case :

I wrote junit test case, if you want you can check data.