

CENG 112 - Data Structures

Assignment 1: The Survival Game

This assignment covers the topics of:

- Strings
- Arrays
- File I/O
- ADTs
- Generics
- Bags

You are expected to implement “The Survival Game” using Java.

There are 4 difficulty levels in this game and the backpack capacity of a survivor decreases as the difficulty increases. The levels include:

- [0] Pilgrim → 9 kg
- [1] Voyager → 7 kg
- [2] Stalker → 5 kg
- [3] Interloper → 3 kg

The backpack capacity is set according to the selected difficulty, at the beginning. Each survivor fills his/her backpack with the items from four different categories (boxes):

- [0] Clothing
- [1] Food & Drink
- [2] First Aid
- [3] Tool

Until the backpack is full, the survivor repeatedly selects the most valuable item from the box with the most items. Since each transfer decreases the number of items in a box, the box from which the item is selected may change continuously. The items in the boxes are listed together in “*items.txt*” file where each line is formed as:

`item_name,item_category_id,item_weight,item_gain`

The *item_gain* is equal to the number of days it prolongs the survivor’s life, and the item with the highest *gain / weight* ratio is assumed to be the most valuable. For example, the line given below represents a can of tuna, which belongs to “Food & Drink” category (*categoryId=1*). It is 0.2 kg and increases the lifespan of the survivor by 2 days. Thus, its value is $2/0.2=10$.

`canned tuna,1,0.2,2`




Your application is expected to perform the following operations:

1. **Create the item boxes:** Read all items from “*items.txt*” file and keep them in separate collections of items by their category. Then, print the number of items and total weight for each box.
(!) You should create an array of boxes and make the box sizes dynamic.
2. **Create the backpack:** Get the difficulty level as 0, 1, 2, or 3 from the user and create an empty collection of items with the corresponding capacity.
3. **Fill the backpack:** Transfer (add and remove) items from the boxes to the backpack.
(!) Box selection must be based on number of items and item selection must be based on value. In case of equality, consider the first candidate as the maximum.
(!!) You cannot exceed the backpack capacity. However, it is not a problem if there is some free space left. That is, you can stop filling when any trial exceeds the capacity.
(!!!) You should also stop when all the boxes are empty.
4. **Display results:** When the filling is done, print the number of items and total weight for each box and the backpack, plus the survivor’s lifespan (the total gain of the items in his/her backpack).
5. **Repeat the steps 2, 3, and 4** until the user inputs **9** or all the boxes are empty.

Sample Application

```
Welcome to Survival Game!
*****
Clothing      10 items | 5.6 kg
Food & Drink  8 items  | 1.9 kg
First Aid     5 items  | 0.6 kg
Tool          9 items  | 5.3 kg
*****
Select Difficulty:
[0] Pilgrim   [1] Voyager   [2] Stalker   [3] Interloper   [9] Exit
0
*****
Clothing      1 items  | 1.2 kg
Food & Drink   1 items  | 0.2 kg
First Aid     1 items  | 0.2 kg
Tool          2 items  | 3.0 kg
Backpack      27 items  | 8.8 kg
Lifespan      115 days |
*****
Select Difficulty:
[0] Pilgrim   [1] Voyager   [2] Stalker   [3] Interloper   [9] Exit
2
*****
Clothing      0 items  | 0.0 kg
Food & Drink   0 items  | 0.0 kg
First Aid     0 items  | 0.0 kg
Tool          0 items  | 0.0 kg
Backpack      5 items  | 4.6 kg
Lifespan      29 days |
*****
No items left in the boxes.
```

Assignment Rules

- This is a group assignment (2 students). However, inter-group collaboration is not allowed!
- All assignments are subject to plagiarism detection and the suspected violations (the solutions derived from or inspired by the solution of other groups) cause to be graded as zero.
- It is not allowed to use Java Collections Framework.
- Your code should be easy to read and test:
 - Keep your code clean. Avoid duplication and redundancy. 
 - Follow Java Naming Conventions. 
 - Use relative paths instead of absolute ones. 

Submission Rules

All submissions must:

- be performed via CMS by only one of the group members,
- be performed before the deadline,
- be exported as an Eclipse Project and saved in ZIP format,
- include all necessary data files (txt, csv, json, etc.) in the right directory,
- follow a specific naming convention such that CENG112_HW1_groupID.

Eclipse Project: CENG112_HW1_G5

Exported Archive File: CENG112_G5.zip

Submissions that do not comply with the rules above are penalized.

NOTE: Before the submission, you must create **a group of two** through the following spreadsheet!

https://docs.google.com/spreadsheets/d/1Zw-4UxPhQXMsEdavHzS8F_OpuLkXuXnm0-Km5K4lu-c