

CSCI 345 - Object Oriented Design

Assignment 01 Part D

Error Handling using Exceptions and Iteration

Overview

This assignment is designed to give you help understanding proper error handling using exceptions and understanding iteration by implementing the Iterable and Iterator interfaces. You can work with one other student on this assignment, or you can elect to complete the assignment on your own.

Program Specification

Mr. Pumphrey received further feedback from Penelope with regards to the Backpack application. Penelope met with the lead developer, Miss Chrysanthemum Crabgrass, and the Backpack application team members that Chrysanthemum leads.

From the discussion Penelope had with Chrysanthemum, it was determined that proper error handling using exceptions rather than Boolean method return values is needed. Chrysanthemum was concerned that too many users of the Backpack class were ignoring the Boolean return values. She was flabbergasted that any sane programmer would ignore a method's Boolean return value! Also, for this revision, Chrysanthemum would like to include iteration support for the Backpack and Pocket classes.

From these discussions, Mr. Pumphrey and Penelope determined that the backpack application should be refactored to support these requirements. A few other requirements have also been determined to benefit the users of the Backpack class. Here's a list of the new requirements:

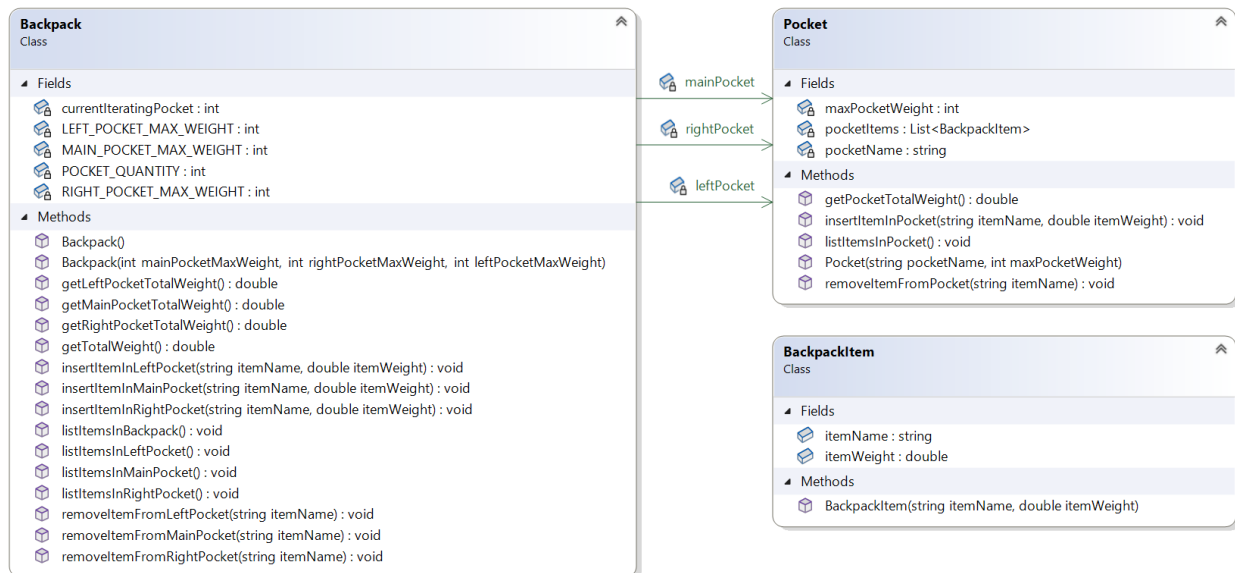
- Create iteration support for the Backpack class.
- Create iteration support for the Pocket class.
- Implement an error and exception handling strategy that uses exceptions rather than Boolean return values to relay error and exception information to Backpack users.
- Add a Backpack constructor that receives three integers representing the maximum weights for the main, right, and left pockets. This constructor enables custom weight limits for the Pocket objects when instantiating a Backpack object.

Error and Exception Handling Specification

- Class/Method: Pocket.insertItemInPocket
 - Example exception messages:
 - Invalid weight exception. Item name: Sleeping bag Item weight: -1.0
 - Weight exceeded exception. Current pocket weight: 7.0 Maximum pocket weight: 10 Item name: Stove Item weight: 4.0
- Class/Method: Pocket.removeItemFromPocket
 - Example exception messages:
 - Item not found exception. Item name: Fork

The UML diagrams for the Backpack and Pocket classes are shown below. A partially completed Backpack class, Pocket class, and test program (Program.java) are uploaded. You can download the files from Canvas->Files-> Assignments-> Assignment_01_D.






Backpack Class UML Diagram



















Backpack

Class

Fields

-  currentIteratingPocket : int
-  LEFT_POCKET_MAX_WEIGHT : int
-  MAIN_POCKET_MAX_WEIGHT : int
-  POCKET_QUANTITY : int
-  RIGHT_POCKET_MAX_WEIGHT : int




Methods

-  Backpack()
-  Backpack(int mainPocketMaxWeight, int rightPocketMaxWeight, int leftPocketMaxWeight)
-  getLeftPocketTotalWeight() : double
-  getMainPocketTotalWeight() : double
-  getRightPocketTotalWeight() : double
-  getTotalWeight() : double
-  insertItemInLeftPocket(string itemName, double itemWeight) : void
-  insertItemInMainPocket(string itemName, double itemWeight) : void
-  insertItemInRightPocket(string itemName, double itemWeight) : void
-  listItemsInBackpack() : void
-  listItemsInLeftPocket() : void
-  listItemsInMainPocket() : void
-  listItemsInRightPocket() : void
-  removeItemFromLeftPocket(string itemName) : void
-  removeItemFromMainPocket(string itemName) : void
-  removeItemFromRightPocket(string itemName) : void






Pocket

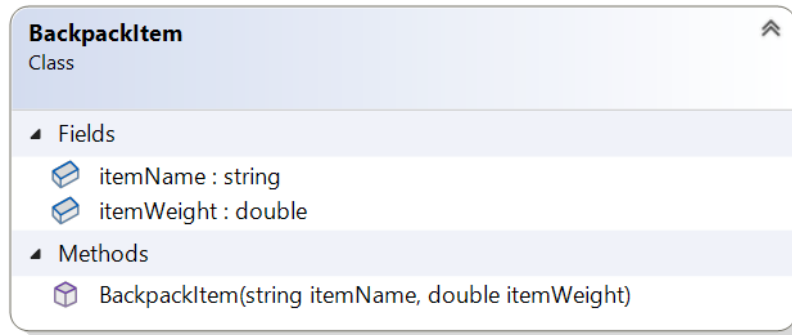
Class

Fields

-  maxPocketWeight : int
-  pocketItems : List<BackpackItem>
-  pocketName : string

Methods

-  getPocketTotalWeight() : double
-  insertItemInPocket(string itemName, double itemWeight) : void
-  listItemsInPocket() : void
-  Pocket(string pocketName, int maxPocketWeight)
-  removeItemFromPocket(string itemName) : void



Program Execution

Below are a few sample program executions:

Example Program Execution

Running the Backpack application...

Backpack total weight: 5.55

Iterating through the backpack:

Item name: Tent Item weight: 5.0
Item name: Cup Item weight: 0.25
Item name: Knife Item weight: 0.3

Example Program Execution

Running the Backpack application...

Invalid weight exception. Item name: Sleeping bag Item weight: -1.0

Example Program Execution

Running the Backpack application...

Weight exceeded exception. Current pocket weight: 7.0 Maximum pocket weight:
10 Item name: Stove Item weight: 4.0

Example Program Execution

```
Running the Backpack application...
```

```
Item not found exception. Item name: Fork
```

Comments

At the top of all source code files, add the following commenting:

```
/*  
# Name:  
# Date:  
# Description:  
*/
```

Submission

Upload the refactored Backpack.java class and the refactored Pocket.java files to Canvas. For this assignment, Canvas has been configured to permit only files that end with the .java file extension.