CS 249 Project 0
Java Review

Overview

This project requires the completion of the following tasks:

- Write and compile programs in Java
- Implement interfaces in Java
- Correctly use Generics in Java
- Apply Unit Testing to your solutions

Interfaces

In this course you will be presented with an interface to implement alongside requirements specifying how the interface must be implemented. Interface names are prefixed with the letter I in this course, to easily distinguish the interface from the classes you may be expected to write. We will start using this convention now, with the interfaces IThing and IBag.

You are provided with an IThing interface which declares the methods required for a class to be considered a thing: getColor(), getMass(), and takeBite(). Starting with the provided files, complete or create the following classes according to the provided comments:

- Apple
- Duck
- QuackException

The desired behaviors for these classes are explained in the comments of the provided files. Apple and Duck declare two new types of things and the QuackException should be thrown when attempting to take a bite out of a Duck object.

Generics

The projects in this class require the use of Java generics, so this part of the homework will give you practice doing just that. You are provided with a generic IBag interface that declares what it means to be a bag. Following the instructions in the comments of the provided code, create a generic Bag class that implements the IBag interface for any type that implements the Thing interface from part 1. While following the provided instructions, you will also create the OutOfSpaceException and the EmptyContainerException classes.

Unit Testing

Lastly, run Unit Tests to validate your solutions to the first two parts of this project, based on the tests in Project0Tests.java. Include in your write up a screenshot showing both the tests passing successfully, and another with at least one test failing.

Start by downloading the interface and unit tests files provided for this assignment and setting-up JUnit for your specific Java development setup. Below are some quick tips for getting started running JUnit tests in several common development setups.

After you turn in your assignment you will be graded using these tests, as well as additional unit tests, so make sure you are thorough in your implementation.

IntelliJ:

- Create a new project and add the interface and test files to the *src* directory
- Open up one of the test files from within IntelliJ and click on one of the lines underlined with an error
- Press ***alt+enter*** (***option+enter*** on mac) and select the option to import the JUnit library into your project
- You should now be able to run each of the tests (***right-click*** the file and choose *run*), or all of the tests (***right-click*** on the project and choose to run all tests) once you create the classes required for this project

Eclipse

- Create a new project and add the interface and test files to the project *src* directory (you might need to ***right-click*** and refresh the *src* icon in Eclipse)
- ***Right-click*** on the project in the Package Explorer panel, go to ***Build Path->Add Libraries***, and select JUnit
- You should now be able to run the unit tests by ***right-clicking*** on the project or test files under Project Explorer, or by going to the ***run->run as*** menu

Command Line

For a command line development environment, you will need to perform several steps:

- Download the required JUnit JAR files to your project directory
- Add the JAR files to your project class path when compiling and running
- Create your own driver class (a class with a main method) to run the tests and print out the results

Write-up and Submission

- Write a short description of the two solutions you provided. In addition, explain what method you chose to perform unit testing, and show screenshots of both tests failing and tests passing using this method.
- Add all of your *.java* files and your write-up document to a .zip or .tar.gz archive and submit it on Bb Learn. Make sure all of your java files are in the root of the zip file, rather than inside additional folders.

- You **must** write and submit your own code. You must also clearly identify and online or text sources that you used as references, any collaborators with which you discussed the project (or lack thereof), and how the source was beneficial.

To Turn In

Turn in these things in a .zip file to BBLearn:

- Source code only (.java files)
    - Do not put your source code in a package
- Write-up
    - This should be in a .docx or .pdf format

DO NOT TURN IN:

- .class files
- interface files
- unit tests
- single screenshots (these should be a part of your write-up)

Grading

This project is worth 25 points.

- 5 Points – Design and code quality
    - Good object-oriented design, clean code, consistent comments, white space, indentation, etc.
- 10 Points – Implementation quality
    - 5 points – Interfaces section
    - 5 points – Generics section
- 10 Points – Write-up and Submission
    - 5 points – Unit Testing screenshots
    - 5 points – Other