Initialization Lab 2

Objectives

1. In this lab students will practice overloading constructors and initializing objects based on constructor arguments.
2. After this lab students should be comfortable manipulating instance variables in constructors and other object methods.

Overview

This lab focuses on initializing objects using constructors and overloading constructors. It also introduces the use of arrays as arguments and instance variables.

Unit Test

You are expected to write unit tests for your code in this lab. Be sure to test the behavior of `ThingContainer` in multiple scenarios, including testing the `add` method when the `ThingContainer is or is not full. You are responsible for determining which other methods need to be tested and how to test them. Remember to practice Test Driven Development as you work on this lab.

Instructions

For this lab, create a java file called `InitializationLab2.Java`. Copy your `ColorfulThing` definition from the last lab.

Part one = parts[0];

Create a new class called `ThingContainer`; this will be a class designed to hold multiple `ColorfulThing`s.

Give the `ThingContainer` class an array of `ColorfulThing`s and a constructor that takes one argument: an integer that defines the size of the array.

`ThingContainer` needs an `add` method to add `ColorfulThing`s to its array. It should add the `ColorfulThing`s in the order they are received. If the array is full when `add` is called then it should print the error message "ThingContainer is full"

`ThingContainer` should also be able to print all of the `ColorfulThing`s it has. Add a printThings method for this.

Create a `main` method for InitializationLab2. In this method create at least three `ThingContainers` and test that you can fill them with randomly generated `ColorfulThing`s and that you get the error message described above.

Part 2: Too many things

Let's enhance our `ThingContainer`. Start by adding a method to remove items. We'll call our method `pop`, and it should remove the last element in the array of things and return that element.

What if we don't want the last element in the array? Let's add a `remove` method, and overload it to handle two different cases. Here they are:

1. We call the `remove` method with a `Color` value from the enumerated type in `ColorfulThing`. Remove the first element of that color from the array and return it. Return `null` if the `ThingContainer` does not contain a `ColorfulThing` of that color.
2. We call the `remove` method with a `ColorfulThing` object. Remove the `ColorfulThing` that matches that object and return it; if it is not in the array return `null`.

Be sure to adjust the elements in the array after each removal so that it stays in order (the next item added should always be added to the end).

Demonstrate the use of all of these new methods in your `main` method.

Part 3:

Add another constructor to `ThingContainer` that can take an array of `ColorfulThings` as its argument. This constructor should initialized its array to match the contents of the array argument.

Modify your `main` method to test that this new constructor works.