Overview: Impossible Change?

Building on a previous activity that asked students to make change using the fewest coins possible (using the greedy solution), students are now asked if it is possible to make change for a certain amount, using a certain number of coins.

Students may find a variety of intuitive ways to solve this problem. The instructor could ask them for their insights into the problem, and use the students' intuitions to draw a search tree for all the possible solutions. They then will be asked to implement the brute-force algorithm for themselves.

This algorithm could be presented in two ways, as an iterative approach or a recursive approach. As a CS2 project, the recursive approach would be better and simpler. To simplify it for WES-CS students, and to prevent potential stack overflow, the iterative approach is preferred here.

Students are invited to use a Linked List (a logical Queue) to hold their progress toward solutions. Even if the students have learned about arrays and how to use them, students may be encouraged by the fact that other Data Structures exist that are easier to use and more suitable to certain applications.

Students may have questions about what makes a Linked List different than an array. It is probably better in this case to emphasize the difference in Interface (a List in general) and not go into details on the Implementation (the LinkedList specifically), particularly noting how easy it is to add and remove items to the collection without worrying about resizing the array. If they have not yet encountered arrays, they probably won't ask these questions.

Prerequisite Material

This activity requires knowledge of:

- ❑ Loops
- ❑ Conditionals

Impossible Change?

Earlier this semester, we looked at ways to make change with the fewest number of coins. This time, we are going to look at a different problem: Is it possible to make change with a certain number of coins?

Part A. First, consider the following problems.
Can you make…

1. $0.50 in ten coins?
2. $1.00 in ninety-three coins?
3. $1.99 in seventy coins?
4. $1.99 in seventy-one coins?

How many different ways can you make change for these amounts:

$0.01 $0.25 $1.00 $1.99 $10.00

How many different amounts can you form with this many coins?

1 5 10

Take some time to discuss as a class how you would approach each of these problems.

Part B. Using the insights you developed in Part A, write a Java program that answers the question: Given integer values x and y, is it possible to make change for *x* cents using exactly *y* coins?

In writing your program, you may find it helpful to use the ChangeCombination class we have provided. A ChangeCombination class represents a handful of coins. You can get or set the number of each denomination of coin, and it will tell you both the total number of coins and the total value of all the coins present. The JavaDoc for this class has been attached.

If you need to keep track of a bunch of ChangeCombinations, use a LinkedList to store your ChangeCombinations. Selected parts of the LinkedList JavaDoc are also attached.

Class ChangeCombination

Represents a combination of coins.

Constructor Summary

ChangeCombination(int quarters, int dimes, int nickels, int pennies)
Creates a ChangeCombination holding the given coins.

Method Summary

int getDimes()
Gets the number of dimes.
int getNickels()
Gets the number of nickels.
int getPennies()
Gets the number of pennies.
int getQuarters()
Gets the number of quarters.
int getTotalNumber()
Gets the total number of coins present.
int getTotalValue()
Gets the total value, in cents, of all the coins present.
void setDimes(int dimes)
Sets the number of dimes.
void setNickels(int nickels)
Sets the number of nickels.
void setPennies(int pennies)
Sets the number of pennies.
void setQuarters(int quarters)
Sets the number of quarters.

Class LinkedList<ChangeCombination>

Linked list implementation of the List interface. This class is a member of the Java Collections Framework.

Constructor Summary

LinkedList<ChangeCombination>()
Constructs an empty list.

Method Summary

boolean add(ChangeCombination e)
Appends the specified element to the end of this list.
void clear()
Removes all of the elements from this list.
ChangeCombination remove()
Retrieves and removes the head (first element) of this list.
int size()
Returns the number of elements in this list.