Lab 6 :: DecoderBot

Due: See Website for details.

**Academic Honesty:** You may work with others to help solve the problem (or sub-problems) but you must write your own code.

**Background:**
Before you start this lab you need to know a few things:

**DefN:** a *char* is a primitive data type that represents one alpha-numeric character. A character is really integer that is looked up in character map. The system of character encodings is called UNICODE which has a integer-to-character mapping for most languages in the world. For English the UNICODE mappings match the ASCII mappings – ASCII was the original English encoding scheme. You will want to look up ASCII in wikipedia for more information about the English int-char mappings.

A variable of type char holds one and only one character. It can be assigned as a character, or as an integer.

**EXAMPLE:**

```java
char foo = 'a'; //note the single quotes for char– double quotes indicate a string
char blah = 97; // makes blah the character code 97, which is a.
```

**DefN:** a *String* is a data type that holds a collection of chars – NOTE: capital S on String. It is NOT a primitive data type, but in many ways it acts like one for convenience. Strings can be composed a number of ways – many of which will look familiar to you from composing println statements.

**EXAMPLE:**

```java
String foo; //declaration of a null string
foo = “Hello there”; //assign some string to the variable.
foo = foo + “, Baker.”; // use ‘+’ to concatenate Strings together.

String blah = “ How are you?”; //another string
foo += blah; // use ‘+=’ to concatenate.

char someChar = 98; // make some character
foo += someChar; // tack it onto the end of foo.

//foo is now = “Hello there, Baker. How are you? b.”

blah = “Hello “ + someChar + “aker, “ + foo; //long concatenations are fine.
```
NOTE on Math:
The Math class that comes with Java has a number of helpful static functions. Among them, Math.pow(int base, int exp), will be particularly useful for this assignment. For example, if you wanted to calculate $2^7$, Math.pow(2, 7) would give you the result.

The Problem:
You encounter a world that has been encoded with beepers. The code represents a string of characters. Each character is encoded by placing (or not placing) beepers in a 7-street column, that spans streets 8-2. An encoded message, therefore, is represented as a series (left to right) of encoded columns that are 7 streets high. (This way a longer message will just extend further to the east where there’s room to expand).

Here is how the column is encoded: Each column represents a number that is the UNICODE/ASCII integer for some character. Each of the 7 streets that make up the column represent a successive power of 2 that should be added to the resulting number. Starting with street 8, a beeper on street 8 adds $2^6$, a beeper on street 7 adds $2^5$, a beeper on street 6 adds $2^4$, …, a beeper on street 2 adds $2^0$. The figure below should add clarity.

Once you’ve calculated a number for the column, you must convert that into the ASCII character it represents and compose a string of all the characters. Your job is to decode an entire message. In decoding your message you must ensure the following:

1.) The original beepers that made up the message must remain where they are (you may pick them up and put them back down if necessary, as long as the original message is there at the end).
2.) You must place a pile of beepers on street 1 underneath each column; the size of the pile is the number represented by the code above it.
3.) You must print the final message as output to the console; it must be the LAST thing printed to the console.

The figure on the next page shows a sample before and after shot for what is expected.
Before: The world as you find it. A message is encoded in the world. The code is 7 streets high and an arbitrary width*

After: Beeper piles, showing the numbers represented in the columns above, are placed on street 1. The last line printed to the console is the decoded message.

Strategy:
You should do some significant planning and thinking about how you want to tackle this. Your life will be significantly easier if you think about using multiple robots to complete simple tasks.

- Identify the sub-problems and smaller tasks that make up the larger problem.
- Think about simple robots you could design (that might use other simple robots) to complete each task. Remember there is no real cost for creating, using, and forgetting about LOTS of robots. (You will see slightly better performance (maybe) if you turnOff a robot when you’re done using it – you may also want to set it invisible so your world doesn’t get cluttered).
- Make one robot at time, test it on a simple set of sub problems (like, make your own worlds to test it), confirm that it works and then move on.
**Turning it in:**
NOTE: See website for due date and time.

On the day that the assignment is due, you will declare what kind of message you want (see below) and I will send you a world file with an encoded message. You will have five (5) minutes to load the world file, decode the message, and do what it says! In other words, your decoder robot better work, and it better work quickly.

**Requirements:**

Part 1 (required):
Being able to decode a 20-character message is the minimum requirement for this assignment. If the characters in message are fewer than 20, it will be padded with spaces.

Part 2 (optional):
Decoding an arbitrarily sized message is slightly harder. In a larger message the length of the message will be the first number encoded in the message. That is, for a long message, the first number will not represent a character but the length of the subsequent message (not including the first number). Obviously, this means a message will max out at 127 characters in length.

Part 3 (optional):
Encode a response message. You may elect to respond to the message I send you by sending back your own encoded message. To do this you will need to clear out the world, encode your message and save the world. Read the World class documentation to see how to do this.