Content Objective: Students will begin to work recursive concepts in Java.

|  |  |
| --- | --- |
| **On the Tech Horizon (10pts.)**  **link to a tech/coding related article or journal no more than one month old (no blogs or reddit clones see below)** | |
| URL: | http://www.bbc.com/news/uk-wales-south-east-wales-38715719 |
| Reaction/Commentary: | *It is amazing what recent technology is capable of. The medical ability of these developments is still limited, but as more research takes place, we could in theory find cures for almost any psychological ailment, or many cranial/cerebral ailments.* |

|  |  |
| --- | --- |
| **Tech Terms and History (20pts.)**  **vocabulary from BJ p.587-628 (definition/commentary/significance in your words)** | |
| Describe the difference between loops and recursion | BJ p.590 Loops are effective for simple functions, as they work from the top down, starting with the smallest values and ending with the largest. Recursion is the reverse. It begins with large quantities and calculates smaller values by plugging in the equation of the next largest function. Recursion is more useful for larger, more complicated functions. |
| What prevents infinite recursion? | BJ p.592 Special tests, such as the getArea method can regulate the function to prevent values from going where they should not, such as negative distances. |
| Strategy for tracing a recursive method in debugger. | BJ p.592-593 Carefully watch the call stack value to verify what recursion you are in |
| What is “trace text” within a recursive method? | BJ p.599-600 calls that show what the value of the method is at during the looping, allowing the user to verify their current value. |
| Describe the call stack. | Glossary and external sources  A list of the methods queued in order the number of times the method is pending because of the recurring function |
| Describe a recursive helper method. | BJ p. 596-597 A method that tests all values from the recursive method |
| Describe permutations of strings and how you would implement them. | BJ p.603-604  Permutations are rearrangements of letters in a string  we loop through all positions in the word to be permuted. For each of them,  we compute the shorter word that is obtained by removing the ith letter |
| Describe the “halt checker” and “killer” in the “Limits of Computation” article | BJ p.606  Halt Checker: keeps track of a value until it reoccurs a given number of time  Killer: an application that runs source code with an applied halt checker |
| Describe mutual recursion | BJ p.608 a method is called by another method during recursion |
| Explain the strategy behind the use of “tokens” in recursion. | BJ p.609-610  Tokens can show you what equation is about to be called on a method. This can show what is about to be done to a function so that you can verify that the next function will have the proper function done to it |
| What is “backtracking”? | BJ p.614-615 A problem solving technique that builds up partial solutions that get increasingly closer to the goal. |
| In the “towers of Hanoi” example, describe how the add method is kept out of an infinite recursion in the Tower class. | BJ resources: “worked example 2”  There is an exception to prevent issues with the values |

|  |  |
| --- | --- |
| **Code Snippets (30pts.)**  **only submit snippets or classes no full programs required (test and run in IDE, then copy/paste applicable code frag)** | |
| Self Check #2 | smallerArea + width + width – 1 |
| Write a recursive Fibonacci method | |  | | --- | | public static long Fibonacci(float i) | |  | { | |  | if (i <= 2) | |  | { | |  | return 1; | |  | } else | |  | { | |  | return Fibonacci(i - 1) + Fibonacci(i - 2); | |  | } | |  | } | |
| Self Check #13 | They are b followed by the six permutations of eat, e followed by the six permutations of bat, a followed by the six permutations of bet, and t followed by the six permutations of bea. |
| E13.3, E13.4, E13.5 |  |
| E13.8, E13.9 |  |
| E13.12, E13 | import java.util.Scanner;  public class Substrings  {    public static void main(String[] args)  {  Scanner sc = new Scanner(System.in);  System.out.println("Computer! Tell me the substring of:");  String str = sc.next();  sbstr(str);  sc.close();  }    public static void sbstr(String word)  {  if (word.length() == 1)  {  System.out.println(word);  return;  } else  {  System.out.println(word);  sbstr(word.substring(0, word.length() - 1));  sbstr(word.substring(1, word.length()));  }    }    } |
| E13.19 |  |

|  |  |
| --- | --- |
| **Code Challenge (30pts.)**  **full functioning application sent to Github** | |
| You may choose any one of the following code challenges: P13.2-P13.6 and post code to GitHub under Recursion. | |
| Notes: |  |

|  |  |
| --- | --- |
| **Badge Progress (10pts.)**  **building your coding profile: Java coding training site to earn badges (recommended site** [**http://coderbyte.com**](http://coderbyte.com) **)** | |
| Screenshot/URL: | ../Screen%20Shot%202017-01-31%20at%209.57.59%20AM.png |
| Notes/Issues: |  |

|  |  |
| --- | --- |
| **Notes**  **your notes** | |
| Notes: |  |