Monkey Business – Practice with Arrays of Objects!

Write a MonkeyZoo class. You will use the Monkey class which is provided below for easy access....

Instance variable – an array of Monkey objects called zoo
Static variable – monkeyId starts at 0

Write a Constructor for your MonkeyZoo class. Create a random number of objects in your zoo. Give each monkey his own idNum based on monkeyId which you increment before assigning.

Create a method void feedMonkeys (int numBan)

For the entire zoo, Check if each monkey is hungry.
If that monkey is hungry, feed him numBan bananas.

Create a method that will find the index of a monkey in the zoo based on number of toys. the method signature is int findMonkey (int nmTys) simply return the index of the first Monkey object with the right number of toys. if there is no monkey with that number of toys, return -1.
/ **
* The Monkey class is an instructional tool for learning.
* about objects and classes.
* @author (Julie Goode)
* @version (1.01 Sept2014)
*/
public class Monkey {

/** instance variables - these are private ENCAPSULATION
//the instance variables can be referred to anywhere within this class,
//but not outside of it. From another class the instance variables can
//only be accessed or changed via the public methods!! */
private String mName;
private int mID;
private String message;
private int numToys;
private boolean isHungry;
private int numBananas;
private Monkey bestFriend;

/** Constructor for objects of class Monkey */
public Monkey(String name, int id) {
    /** initialise instance variables, those that are not
* instantiated will be defaulted by the JVM;
* int gets 0, String gets null, boolean gets false */
mName=name;
isHungry=true;    //monkeys are always hungry!!
mID = id;
}

/** document comment
* @param   msg   holds a string variable
* @return none because void or mutator method
*/
public void setMessage(String msg) {
    message = msg;
    //note: if I put msg=msg; then the instance variable is never set
    //msg which is a local variable is set to itself. nothing happens!!
}

/**
* @param none we are making the monkey speak
* @return none because void or mutator method
*/
public void speak() {
    System.out.println(message);
}
/**
 * @param   n  integer number of toys to give monkey to play with
 * @return  none  because void or mutator method
 */
public void addToys(int n) {
    numToys += n;
}

/**
 * @param   none      we are asking the monkey if it is hungry
 * @return  boolean   return true if our monkey is hungry
 */
public boolean isHungry( ) {
    if (numBananas > 3) {
        isHungry = false;
    } else {
        isHungry = true;
    }
    return isHungry;
}

/**
 * @param   n      integer number of bananas to give monkey to eat
 * @return  none   because void or mutator method, alters numBananas
 */
public void eatBananas(int n) {
    int total = n + numBananas;
    numBananas = total;
}

public String getMsg() {
    return message;
}

public String getName() {
    return mName;
}

/**
 * @param   none      we are asking the monkey how many toys it has
 * @return  boolean   return true if our monkey is hungry
 */
public int howManyToys( ) {
    return numToys;
}
/** Note - it is convention that places the constructors at the front of the class. The code compiles even if this is at the end of the code... */

public Monkey(int numT, String mName) {
    this(mName);
    // isHungry = true;
    numToys = numT;
    message = "ooh ooh ooh ooh ooh!";
}

@Override
public String toString() {
    return ("Monkey: \n    Hungry? \n    Has how many toys: " + howManyToys() + 
    Message: " + getMsg() + 
    \n\n    ");
}