Monkey Business – Practice with Objects and Classes!

Write a MonkeyRunner class with a main method in it. You will use the Monkey class which is provided on our web site under “video Learning”.

It is also provided below for easy access….

Instantiate a Monkey object. Name your monkey “Bob”. The identifier can be anything you like. Using that monkey instance – do the following:

Feed your monkey 3 bananas.

Check if he is hungry.

If he is hungry, feed him 3 more bananas.

Check if he is hungry again.

If he is, feed him 3 more bananas.

If he is not hungry, then give him 2 toys to play with.

Display (means use System.out.println) the number of bananas he has eaten and how many toys he has.

Display the monkeys message with his name

Set his message to a cool slogan.

Display his message again, your message should show up!

/\*\*

 \* 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 String message;

 private int numToys;

 private boolean isHungry;

 private int numBananas;

 private Monkey bestFriend;

 /\*\* Constructor for objects of class Monkey \*/

 public Monkey(String name)

 {

 /\*\* 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!!

 }

 /\*\* 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 veriable 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;

 //note: we did not set the numToys instance variable but the JVM will

 //default int variables to 0. win-win!

 }

 public void setBest(Monkey m) {

 bestFriend = m;

 System.out.print(mName+" has a best friend named "+ m.mName);

 }

 /\*\*

 \* @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 > 5)

 isHungry = false;

 else

 isHungry = true;

 return isHungry;

 }

 /\*

 public int isHungry() {

 return numBananas;

 } \*/

 /\*\*

 \* @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){

 // note: total is a local variable. only accessed in this method!

 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 oooh ooh!";

 }

 @Override

 public String toString() {

 return ("Monkey: "+getName()+"\nHungry? "+isHungry()+"\nHas how many toys: "+

 howManyToys()+"\nMessage: "+getMsg()+"\n");

 }

}