CS125 Section 8 "I Object!" SOLUTION

Conjecture	True for Arrays?	True for Object?
1 Holds multiple pieces of data.	Y	Y
2 Data must be of the same type.	Y	N
3 Can compare contents to another of same type using ==	N	N
4 Contents can be copied by using =	N	N
5 Makes a new object when passed as a parameter to a function.	N	N
6 Allows programmers to build new types.	N	Y
7 Internal elements are accessed using the [] or "square bracket" operator.	Y	N
8 Internal elements are accessed using the . or "dot" operator.	N	Y

When we define a class, we put two kinds of things in a class file: methods and data. **Methods** are code *e.g.,* Math.random() which returns a new random number for us, and **values** are variables that hold state *e.g.,* Math.PI which holds an approximation of π. Calling a method uses parentheses. Accessing a value does not. A method is a class method (aka static method) or an instance method.

```
public class Holiday {
 public static String[] months = {"Jan", "Feb", "Mar",..., "Aug", "Sep", "Oct", "Nov", "Dec"};
 static int numHolidays;   // keeps track of how many holidays have been created
 String name;
 int month, day;
 public static Holiday createNewHoliday(String theName, int theMonth, int theDay) {
  numHolidays ++;   // we're creating a new Holiday!
  Holiday result = new Holiday();
  result.name = theName;
  result.month = theMonth;
  result.day = theDay;
  return result;
 }
 public static int getNumHolidays() {
                        return numHolidays;}
 public String getName() {return this.name;}
 public int getMonth() {return this.month;}
 public int getDay() { return this.day;}
 public Holiday mystery1() {
  Holiday h = new Holiday();
  h.name = this.name;
  h.month = this.month;
  h.day = this.day;
  return h;
 }
public boolean mystery2(Holiday other) {
  return this.name.equals(other.name) &&
         this.month == other.month &&
         day == other.day;
 }
}
```

	Class(static)	Instance	Local (temporary)
Value	numHolidays	name month day	h other result
Method	getNumHolidays createNewHoliday	mystery1 mystery2 getName getMonth getDay	Not Applicable

1. Use the above grid to classify all of the methods and values which are local variables.
2. Explain why the class would *still* compile if the programmer forgot the 'static' for *`getNumHolidays`*.

Can always access class(static) variables from an instance method (but now programmers using the Holiday class need a reference to a Holiday object to execute getNumHolidays method)

3. Explain why the class would *not* compile if the programmer added static to *`getName`*?

Method accesses "name" which is an instance variable. There is no 'this' in a class method.

4. Choose better names for *`mystery1`* and *`mystery2`* methods.

copy, equals.

5. Why did the programmer use `name.equals` but '==' to compare `month` and `day`?**Compare two strings, two holidays may use two different Java String objects with the same character sequence.**

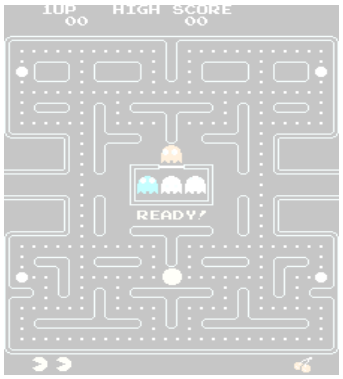
Calling Class(Static) vs. Instance methods

1. For each line of code below, identify whether the method call is invoking instance methods or class methods of the Holiday object or class, respectively.
2. If the main method had been written inside the Holiday class, how could this code be written shorter?

```
public class HolidayRunner {  // Notice we're inside a different class
 public static void main(String[] args) {
   Holiday hween = Holiday.createNewHoliday("Halloween", 10, 31);                  CLASS
   TextIO.put("My favorite holiday is " + hween.getName());                        INSTANCE
   TextIO.put(" which is on " + Holiday.months[hween.getMonth() - 1]);             INSTANCE
   TextIO.putln(" " + hween.getDay());                                             INSTANCE

   Holiday lincoln = Holiday.createNewHoliday("Lincoln's Birthday", 2, 12);        ClASS
   if (! hween.mystery2(lincoln)) {                                                INSTANCE
     TextIO.putln("Created " + Holiday.getNumHolidays() + " different ones");      CLASS
   }
   Holiday halloween = hween.mystery1();                                           INSTANCE
   Holiday lincolnsBday = lincoln;
 }
}
```

3. Object References are just pointers (aka "Zombies!") not actual objects. Explain why the following code only creates three ghost objects.
4. What is printed by the last line of the following Java code? true or false? **true**
5. Which objects are no longer referenced (Blinky Pinky Inky) and can be recycled by the garbage collector?




```
Ghost g1, g2, g3, g4;
g1 = new Ghost();  // Blinky
g2 = new Ghost();  // Pinky
g3 = new Ghost();  // Inky
g4 = g2;
g1 = g3;  BLINKY NO LONGER REFERENCE NOW g1 is reassigned
g2 = g3;
g3 = g2;
boolean result = (g3 == g1);
System.out.println(result);
```

6. SPOT AND FIX THE MISTAKES.

```
class DodgyDice {
  private int side=0; // 0...5

  public int roll() {
     side = (side + 1) % 6;
     return 1 + side;
  }
  public static boolean rolledSix() {
    return side == 5;
  }
}


Example use
DodgyDice d6 = new DodgyDice () ;
int rick = d6.roll();
TextIO.putln("I rolled " + rick);
if(d6.rolledSix()) TextIO.put("Lucky");
```

```
class QuoteList {
  private String[] array = new String[1000];
  private int used = 0;
  public void add(String quote) {
// I'll show you how to improve on this in lecture
     this.array[ used++ ] = quote;
  }
  public int countEmpty() {
    int result = 0;
    for(int i =0;i< used;i++)
        if(array[i].length() ==0 ) result ++;
    return result;
  }
  public boolean equals(QuoteList other) {
    if(other.used != this.used) return false;
    for(int i =0;i< used ;i++)
       if( ! array[i].equals(other.array[i]))
         return false;
    return true;
  }
}
```