package lab1;
import java.util.Random;
class RandomHello {
    /**
     * @effects uses a RandomHello object to print
     * a random greeting to the console
     */
    public static void main(String[] argv) {
        RandomHello randomHello = new RandomHello();
        System.out.println(randomHello.sayHello());
    }
    /**
     * @return a random greeting from a list of five different greetings.
     */
    public String sayHello() {
        String[] greetings = new String[5];
        gs[0] = "A hint for the lab!";
        // YOUR CODE HERE
    }
}

boolean: true, false
byte: -128 to 127
short: -2¹⁵ to 2¹⁵-1
int: -2³¹ to 2³¹-1
long: -2⁶³ to 2⁶³-1
double, float: reals, ie 3.14159, 100000e-9, 400.02

program structure:
package - location of java file in src folder
import - to use Library classes, such as Random
class RandomHello - template for making objects
public static void main(String[] argv) - entry point; program starts execution here
(static = single, separate to objects, part of class template)
public String sayHello() - a method of RandomHello
RandomHello randomHello = new RandomHello() - create new RandomHello instance
String[] greetings = new String[5] - create an array of 5 Strings
  greetings[0] = "foo" - set the first String in the array to "foo", syntactic sugar for new String("foo")

primitives
<table>
<thead>
<tr>
<th>char: 'a', 'b', ..., 'A', €, ¢, ¶</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>final static int EMPTY_SQUARE = 0;</code></td>
</tr>
</tbody>
</table>

**Constant**
- `final` = cannot change
- `static` = single variable shared by all instances, part of class template

```
char[] alphabet = new char[26];
alphabet[0] = 'a';
alphabet[25] = 'z';
```

**Arrays**
- `String[] myPets = new String[] {"Fluffy", "Muffy", "Scruffy");`  
- `int[][] chessBoard = new int[8][8];`

**Initialize when constructing**
- Initialize by looping, two dimensional array

```
for (int row = 0; row < chessBoard.length; row++) {
    for (int col = 0; row < chessBoard[0].length; col++) {
        chessBoard[row][col] = EMPTY_SQUARE;
    }
}
```

**For loop**
- `i = 0, 1, ... , 8, 9`

```
for (int i = 0; i < 10; i++) {
    //do something here;
}
```

**While loop**
- while predicate evaluates to true, keep doing something (that something must make predicate false eventually, or else you have an infinite loop!)

```
while (predicate) {
    //do something here;
}
```