Aufgabe 1: Implementation "direktes Auswählen" (siehe ALG4 Slide 30ff.)

Implementieren Sie eine Klasse `Sort` mit einer Methode `directSelect()`.
Testen Sie Ihre Methode. Vielleicht geht's hier schneller ohne JUnit, z.B. mit einer Ausgabe auf die Konsole.

```
/**
 * SelectionSort
 *
 * @author Christian Bontekoe, Felix Rohrer
 * @version 1.0
 */
public class SelectionSortWiki
{

    /**
     * Constructor for objects of class SelectionSortWiki
     */
    public SelectionSortWiki()
    {

    }

    /**
     * Sort an array of int with "selectionsort"
     *
     */
    public void directSelect()
    {
        int[] a = {44, 87, 40, 20, 74, 49, 10, 85, 73, 8, 25, 46, 70, 32, 56, 27, 61, 30, 19, 30};

        /* a[0] to a[n-1] is the array to sort */
        int iPos;
        int iMin;
        int iTmp;

        /* advance the position through the entire array */
        /* (could do iPos < n-1 because single element is also min element) */
        for (iPos = 0; iPos < a.length; iPos++) {
            /* find the min element in the unsorted a[iPos .. n-1] */

            /* assume the min is the first element */
            iMin = iPos;
            /* test against all other elements */
            for (int i = iPos+1; i < a.length; i++) {
                /* if this element is less, then it is the new minimum */
                if (a[i] < a[iMin]) {
                    /* found new minimum; remember its index */
                    iMin = i;
                }
            }

            /* iMin is the index of the minimum element. Swap it with the current position */
            if ( iMin != iPos ) {
                iTmp = a[iPos];
                a[iPos] = a[iMin];
                a[iMin] = iTmp;
            }
        }

        // print out
        for (int i = 0; i < a.length; i++) {
            System.out.println(a[i]);
        }
    }
}
```

Aufgabe 2: Analyse "direktes Auswählen"

Analysieren Sie den Sortieralgorithmus "direktes Auswählen". Konsultieren Sie dazu nochmals die Unterlagen. Gehen Sie analog wie beim "direkten Einfügen" vor.
done

Aufgabe 3: Iteration vs. Rekursion, JUnit

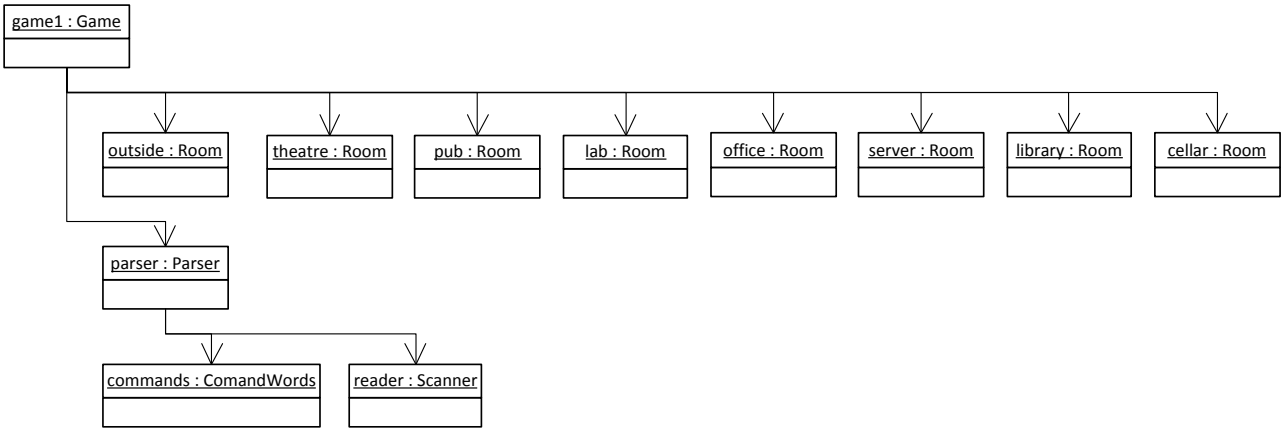
Implementieren Sie eine Klasse `Multiplication` mit den beiden Methoden `multIt()` und `multRe()` (siehe Vorlesung). Testen Sie die Methoden mit Hilfe von JUnit.

```
/**
 * Iteration vs. Rekursion
 *
 * @author Christian Bontekoe, Felix Rohrer
 * @version (1.0
 */
public class Multiplication
{

    /**
     * Constructor for objects of class Multiplication
     */
    public Multiplication()
    {

    }

    public int multIt(int a, int b) // a > 0, b > 0
    {
        int result = b;
        for(int i = 1; i < a; i++) {
            result += b;
        }
        return result;
    }

    public int multRe(int a, int b) // a > 0, b > 0
    {
        if(a == 1) { // Rekursionsbasis
            return b;
        }
        else { // Rekursionsvorschrift
            return multRe(a-1, b) + b;
        }
    }
}


/**
 * The test class MultiplicationTest.
 *
 * @author Christian Bontekoe, Felix Rohrer
 * @version (1.0
 */
public class MultiplicationTest extends junit.framework.TestCase
{
    /**
     * Default constructor for test class MultiplicationTest
     */
    public MultiplicationTest()
    {
    }

    /**
     * Sets up the test fixture.
     *
     * Called before every test case method.
     */
    public void setUp()
    {
    }

    /**
     * Tears down the test fixture.
     *
     * Called after every test case method.
     */
    public void tearDown()
    {
    }

    public void testIterationAndRekursion()
    {
        Multiplication multi = new Multiplication();
        assertEquals(multi.multIt(3, 5), multi.multRe(3, 5));
    }
}
```

Aufgabe 4: Objektdiagramm

Lösen Sie die Aufgabe 7.12 aus dem Lehrbuch.

7.12:



Optionale Zusatzaufgabe 5: Refactoring

Lösen Sie die Aufgabe 7.18 aus dem Lehrbuch.

7:18

```
public class Game
{
    [...]

    /**
     * Print out some help information.
     * Here we print some stupid, cryptic message and a list of the
     * command words.
     */
    private void printHelp()
    {
        System.out.println("You are lost. You are alone. You wander");
        System.out.println("around at the university.");
        System.out.println();
        System.out.println("Your command words are:");
        System.out.println(parser.getCommands());
    }
}


public class Parser
{
    [...]

    /**
     * Return a list of valid command words.
     * @return Command List
     */
    public String getCommands()
    {
        return commands.getCommandList();
    }
}


public class CommandWords
{

    [...]

    /**
     * Returns all valid commands
     * @return Commandlist
     */
    public String getCommandList()
    {
        String res = "";
        for(String command : validCommands) {
            res += command + "   ";
        }
        //.trim -> remove spaces at the end
        return res.trim();
    }
}
```