Code Review Exercise

For this exercise you need to sit with your classmates and evaluate the code they wrote for the ClearCellGame and WebPageGenerator project. This is what you need to do for this exercise:

1. Form groups of three students.

2. Taking turns, one student will show the implementation of a project to the other members of the group. Students will provide feedback by going throw each of the items of the "Style Checklist" you will find below.

3. In addition, look at the following items:

   a. Is there duplicate code that could have been implemented in an auxiliary method?
   b. Mention anything that can improve the clarity of the code or that could have simplify the implementation.
   c. IMPORTANT: Provide constructive criticism; respect is extremely important.

4. Notice this is not a graded exercise.

Style Checklist

1. Good names for variables, constants, and methods. Names should be descriptive and use camelCase.
2. Consistent style for curly brackets (pick one style and stick with it). For example:

```
/* valid style */
if (x > y) {
  m = 10;
     } else {
  m = 200;
}
```

```
/* valid style */
if (x > y)
{
  m = 10;
}
else
{
  m = 200;
}
```

3. Use braces; avoid loops and conditionals without them.
4. Consistent indentation (2, 3, 4, etc. spaces). Remember to always use the same number of spaces. Be careful with the tab key as your code indentation may appear different when using different editors.
5. In your code you should leave one blank space between binary operators (e.g., x = 5 + 7).
6. About return statements
   - Return at the end of a method is OK.
   - Return at the beginning of method for erroneous input is OK.
7. You should leave one blank line between variable declarations and code. For example:

```
int temperature, distance;

System.out.println("...");
```

8. Make sure you define values as "symbolic constants" when needed. Do not use numbers (literals) in your expressions if they have special meanings. For example: Instead of using the literal "3.1415927" in an expression, define and use a symbolic constant like this:

```
public static final double PI = 3.1415927;
```

9. You should avoid long lines of code (keeping them around 80 characters is recommended).
10. The number of lines of your code can be reduced by defining variables of the same type in the same line. For example:

```
int x;
int y;
int w;
float a;
float b;
float c;

/* alternative */
int x, y, w;
float a, b, c;
```

11. Using i, j, k as the name of the iteration variable of a for loop is fine although in some cases a descriptive name can help (e.g., a nested loop to process a 2-dim array can benefit from using row and col).
12. Using break is OK, but do not abuse it.
13. Using continue is OK, but do not abuse it.
14. Do not call static methods by using a class instance.
15. Remove from your code variables that are declared but never used.
16. Remove any unnecessary import statements.
17. You must avoid code duplication by calling appropriate methods (rather than cutting and pasting code). You may define your own private utility methods to perform often repeated tasks.
18. Make sure you recognize when a field should be defined as private, protected, public, or package.
19. The clarity of code can benefit from using empty lines between segments of code. Make sure you take this into account while writing your code. Similarly, comments in your methods are important in order to clarify what your code is doing. Make sure you provide such comments when needed. Methods (at the top) should include a description of the task performed by the method.