Introduction to Object-Oriented Programming: Project

Practical Guidelines

- Students work on the project **individually!** Code duplication between students will result in severe penalties for all parties involved; no code copying from online sources allowed.

- Questions regarding the project assignment should be asked in **the Toledo forum (Assignment Questions)** or during the exercise sessions. Only these questions will be answered, **no questions will be answered via e-mail**; we will not help with technical questions (i.e., fixing bugs in your code).

- The deadline for the final project submission is **Thursday, December 30th 2021, 23:59**. No projects received after this date will be accepted.

- The use of JavaFX is strictly forbidden as the course is focused on Java, and hence **only Java code is allowed** for all aspects of the course; **submissions that contain any amount of JavaFX code will not be graded!**

- An electronic version of your project should be submitted via Toledo, via the Assignments module. As discussed during the first class, multiple submissions will be allowed. However, Toledo should not serve as your daily backup service, and hence **the number of submissions allowed will be restricted to 5! No exceptions** will be made (regardless of any reasons you might have); the idea is that you upload your project code when it's complete and fully functional, but you can still submit corrections in case you find a bug or another type of problem. Projects that are late will be discarded; **the final submission will be used for grading** (again, **no exceptions**).

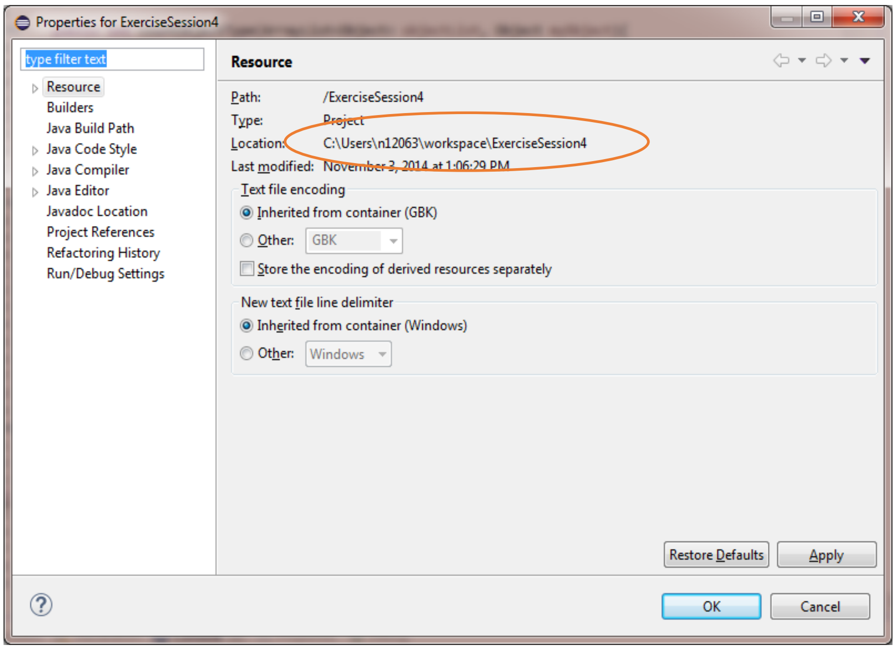
- The Toledo submission system will indicate a mark of 10 as the system requires a maximum mark to be provided. As indicated in the introductory presentation for the course, the Java project counts for 10 marks out of 20 for students of the Master of Artificial Intelligence and the Master of Information Management, and for 7,5 marks out of 20 for all other students (i.e., those that also take the R part of the course; hence the marks will be rescaled).

- You must include the following components in your project submission, preferably in a zipped file with **surname and first name** as the name of the file: **surname_firstname.zip:**
  - Java project source code copied from your Eclipse / IntelliJ workspace (i.e., the src folder), including all the required images (when applicable) for your project to run in the correct location; no .class files or .jar files should be submitted, do not include the /bin directory; your project should contain exactly **one main method**.
  - Short report describing your project (**maximum** 4 pages, see guidelines) with strong focus on the structure and design of your code; no UML diagrams should be provided.

- Prepare your report and code (names, comments, etc.) in **English**.

- To copy your source code from Eclipse:

    - From Eclipse, right click on your project, select Properties. Under Resource, you will find the location of the project folder.
    - Navigate to this location on your computer and include a copy of the src folder in your project in your submission.



Resource

Path: /ExerciseSession4
Type: Project
Location: C:\Users\n12063\workspace\ExerciseSession4
Last modified: November 3, 2014 at 1:06:29 PM

Text file encoding
[✓ Inherited from container (GBK)]
Other: GBK
Store the encoding of derived resources separately

New text file line delimiter
[✓ Inherited from container (Windows)]
Other: Windows

- To copy your project from IntelliJ:

    - In IntelliJ, you can see the location of your project in the Project view/panel (next to the project name that's listed in bold).

- It's a good test to see if you can copy your project to another computer and see if you can run it there from within an IDE such as Eclipe, IntelliJ or Netbeans. Most programming environments have the possibility of "Creating a Project from Existing Local Sources".

- Another good test is to try to run your application on a different operating system than the one you used during the development process.

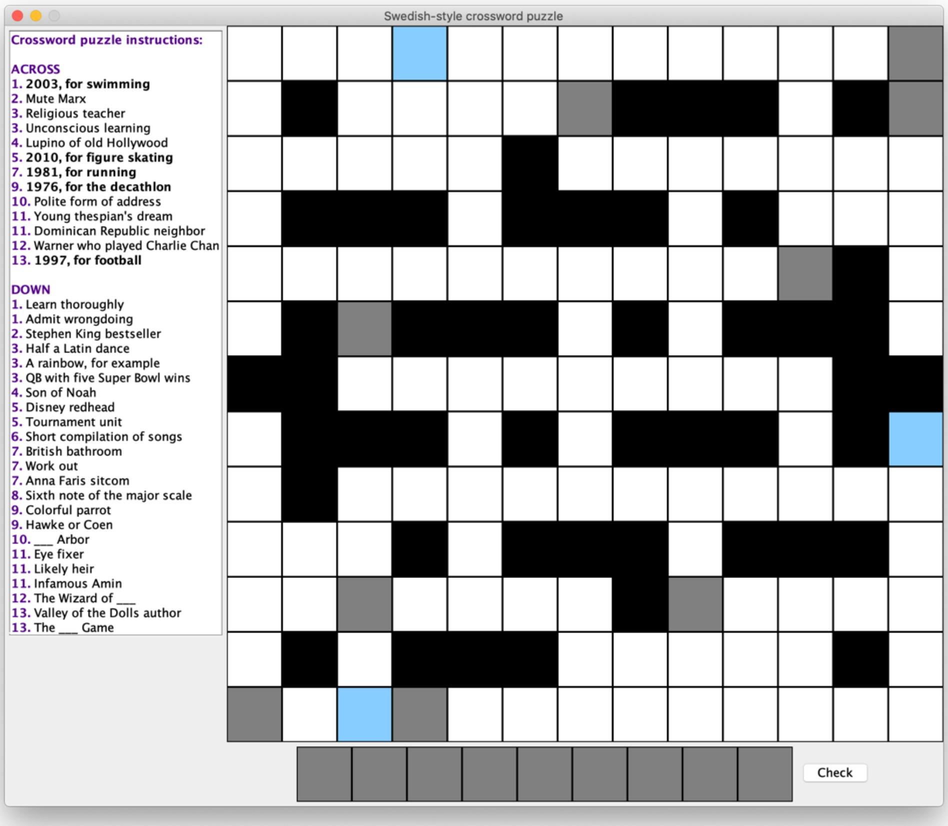
Coding Guidelines

The code guidelines below are a selection from a more extensive set of guidelines provided by Oracle (and can be found on the oracle website). We expect you to follow at least these guidelines:

- Use a separate file for each class and interface
- Assign classes that are not related to each other to different packages
- Write at most 1 statement per line
  `int x = 5;` *//(good)*
  `int y = 10;`
  is preferred over
  `int x = 5; int y = 10;` *//(bad)*
- Naming conventions
  - **Classes**
    - Class names should be nouns, in mixed case with the first letter of each word capitalized. Try to keep your class names simple and descriptive. Use whole words—avoid acronyms and abbreviations (unless the abbreviation is much more widely used than the long form).
      - Start with upper case, e.g.: `Person`, `Car`, `Game`, `BMICalculator`
  - **Interfaces**
    - Start with capital letter, e.g.: `Nameable`, `Capable`, `Pettable`
  - **Methods**
    - Names are usually verbs, and should start with lower case and use camelCase if it consists of multiple words, e.g.: `getValue()`, `add()`, `performSomeAction()`
  - **Variables**
    - Variable names should be short (where possible) yet **meaningful**!. Choose variable names which indicate to the casual observer the intent of its use. One-character variable names should be avoided except for temporary "throwaway" variables. Common names for temporary variables are: `i`, `j`, `k`, `m`, and `n` (for `int`) and `c`, `d`, and `e` (for `char`).
    - Name starts with lower case and use camelCase if it consists of multiple words, e.g.: `name`, `dateOfBirth`, `ageLimit`, `stomachContent`
  - **Constants**
    - Should be all uppercase, use "_" to separate words, e.g.: `PI`, `DAYS_IN_WEEK`, `GRADE_TO_PASS`
- Provide comments for each variable, method, and class so that it is 100% clear to the reader of your code what its purpose is.
- Use proper indentation for your code. Every time you start a block of code (when making a class/method/if-else if-else/switch/…) indent the code within the block.
  - TIP: Eclipse and IntelliJ can do this for you.
    Eclipse: Select the code you want to indent properly > right-click > source > Correct Indentation
    IntelliJ: Select the code, Code > Auto-Indent Lines

Project Guidelines

For this project, you need to create an implementation of a crossword puzzle, a single-player puzzle (or game) that typically has the form of a square or a rectangular grid of white- and black-shaded squares. There are different names such a puzzle goes by, including a Swedish-style crossword puzzle. Importantly, the rules of the intended game are explained in this document, and you should not copy different rules or instructions from (types of) puzzles you can find online. However, it is often a good idea to think how you could extend or modify your code to attain the functionality of a different version of the puzzle proposed here.



Crossword puzzle instructions:

ACROSS
1. **2003, for swimming**
2. Mute Marx
3. Religious teacher
3. Unconscious learning
4. Lupino of old Hollywood
5. **2010, for figure skating**
7. **1981, for running**
9. **1976, for the decathlon**
10. Polite form of address
11. Young thespian's dream
11. Dominican Republic neighbor
12. Warner who played Charlie Chan
13. **1997, for football**

DOWN
1. Learn thoroughly
1. Admit wrongdoing
2. Stephen King bestseller
3. Half a Latin dance
3. A rainbow, for example
3. QB with five Super Bowl wins
4. Son of Noah
5. Disney redhead
5. Tournament unit
6. Short compilation of songs
7. British bathroom
7. Work out
7. Anna Faris sitcom
8. Sixth note of the major scale
9. Colorful parrot
9. Hawke or Coen
10. ___ Arbor
11. Eye fixer
11. Likely heir
11. Infamous Amin
12. The Wizard of ___
13. Valley of the Dolls author
13. The ___ Game

The game's goal is to fill the non-black squares with letters, forming words, names (of locations or people) or phrases, by solving clues or instructions, which lead to the answers. In languages that are written left-to-right, the answer words and phrases are placed in the grid from left to right ("across") and from top to bottom ("down"). The black squares are used to separate the words or phrases. As you can see from the screenshot above, the clues / instructions are shown on the left-hand side of the application. You are provided with such an example file as part of the project assignment. Important to note is that the file provided contains simple text and hence not any information on which text appears in color or bold, but you can derive these from the input file structure (but see page 6 for more information). The input file used for the puzzle shown in this assignment is shown on the next page in this document. Note that there are no single-square answers to the clues / instructions, i.e., all answers comprise at least 2 squares.

```
13 13
O O O H(H) O O O O O O O O S
O X O O O O S X X X O X S
O O O O O X O O O O O O O
O X X X O X X X O X O O O
O O O O O O O O O O S X O
O X S X X X O X O X X X O
X X O O O O O O O O O X X
O X X X O X O X X X O X H(C)
O X O O O O O O O O O O O
O O O X O X X X O X X X O
O O S O O O O X S O O O O
O X O X X X O O O O O X O
S O H(Y) S O O O O O O O O O
ACROSS
1. 2003, for swimming
2. Mute Marx
3. Religious teacher
3. Unconscious learning
4. Lupino of old Hollywood
5. 2010, for figure skating
7. 1981, for running
9. 1976, for the decathlon
10. Polite form of address
11. Young thespian's dream
11. Dominican Republic neighbor
12. Warner who played Charlie Chan
13. 1997, for football
DOWN
1. Learn thoroughly
1. Admit wrongdoing
2. Stephen King bestseller
3. Half a Latin dance
3. A rainbow, for example
3. QB with five Super Bowl wins
4. Son of Noah
5. Disney redhead
5. Tournament unit
6. Short compilation of songs
7. British bathroom
7. Work out
7. Anna Faris sitcom
8. Sixth note of the major scale
9. Colorful parrot
9. Hawke or Coen
10. ___ Arbor
11. Eye fixer
11. Likely heir
11. Infamous Amin
12. The Wizard of ___
13. Valley of the Dolls author
13. The ___ Game
```

Regarding how to ensure that different parts of the text are shown in different fonts, you can have a look at the official Java tutorial on "Using Text Components": https://docs.oracle.com/javase/tutorial/uiswing/components/text.html

As you can read there, styled text components are powerful and flexible, with a JTextPane as a popular example: https://docs.oracle.com/javase/8/docs/api/javax/swing/JTextPane.html

Description of the input file / puzzle:

The theme of this particular crossword puzzle is sports. Every year, the James E. Sullivan Award is presented to the USA's most outstanding amateur athlete. Can you deduce the identities of five other winners **with just the year and the sport as clues**? As you can see in the example screen shots, puzzles that have a year as the first part of the clue / instructions use a font that highlights these clues in the left-hand side of the application, in bold.

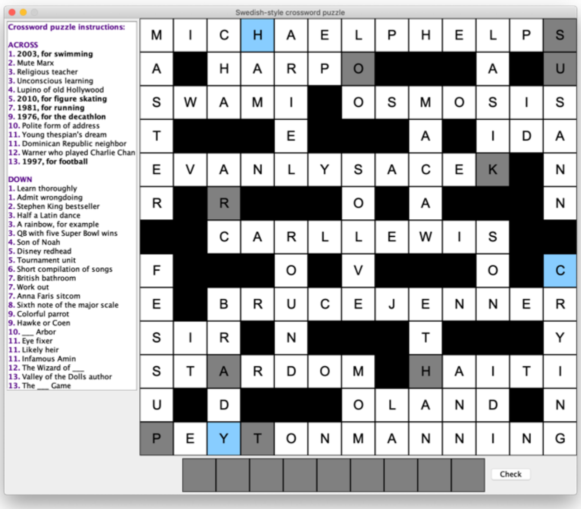
The first line of the input file contains the dimensions, i.e. the number of rows and columns of the playing field. Note that the playing field does not have to be a square. On the following lines, each row of the playing field is described by a character for each type of square (see the next page for more information on the square types):

- O: regular (white) square
- X: black square
- S: special square
- H(…): help square, containing between parentheses the correct character to be entered onto the playing field at this position; for example H(C) means that the square should hold a 'C' as the correct character

All the other remaining lines in the input file contain the clues or instructions for the player of the puzzle, i.e., which solutions / answers need to be provided horizontally (across, the rows are numbered from top to bottom) and vertically (down, the columns are numbered from left to right). When more than one answer per line is required, the row and / or column number will be used multiple times, meaning that the first clue needs to be answered in the first set of available squares, followed by answers to the second, third, … clues on the remaining squares.

For example, for the first line <u>across</u> we're looking for the winner of the James E. Sullivan Award in 2003, for swimming. The winner was Michael Phelps (the answer does not contain any spaces). For the 11th line across, there are two clues (11. Young thespian's dream & 11. Dominican Republic neighbor). As you can see on the playing field, there is room for two answers: the first answer consists of 7 characters / squares (i.e., stardom), and the second answer consists of 5 characters / squares (i.e., Haiti). Similarly, for the first line <u>down</u> you are provided with two clues (1. Learn thoroughly & 1. Admit wrongdoing), as there is room for two answers, both consisting of 6 characters / squares each (i.e., master & fessup).

On the next page, you can see the complete solution of the puzzle.



Crossword puzzle instructions:

ACROSS
1. **2003, for swimming**
2. Mute Marx
3. Religious teacher
3. Unconscious learning
4. Lupino of old Hollywood
5. **2010, for figure skating**
7. **1981, for running**
9. **1976, for the decathlon**
10. Polite form of address
11. Young thespian's dream
11. Dominican Republic neighbor
12. Warner who played Charlie Chan
13. **1997, for football**

DOWN
1. Learn thoroughly
1. Admit wrongdoing
2. Stephen King bestseller
3. Half a Latin dance
3. A rainbow, for example
3. QB with five Super Bowl wins
4. Son of Noah
5. Disney redhead
5. Tournament unit
6. Short compilation of songs
7. British bathroom
7. Work out
7. Anna Faris sitcom
8. Sixth note of the major scale
9. Colorful parrot
9. Hawke or Coen
10. ___ Arbor
11. Eye fixer
11. Likely heir
11. Infamous Amin
12. The Wizard of ___
13. Valley of the Dolls author
13. The ___ Game

Description of the playing field and interactions:

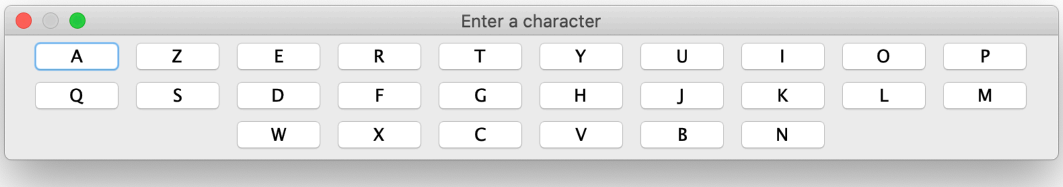
The main playing field on the right-hand side of the application contains all the squares and the puzzle's solution field along with a 'Check' button. As you can see from the provided screenshots, there are 4 different types of squares:

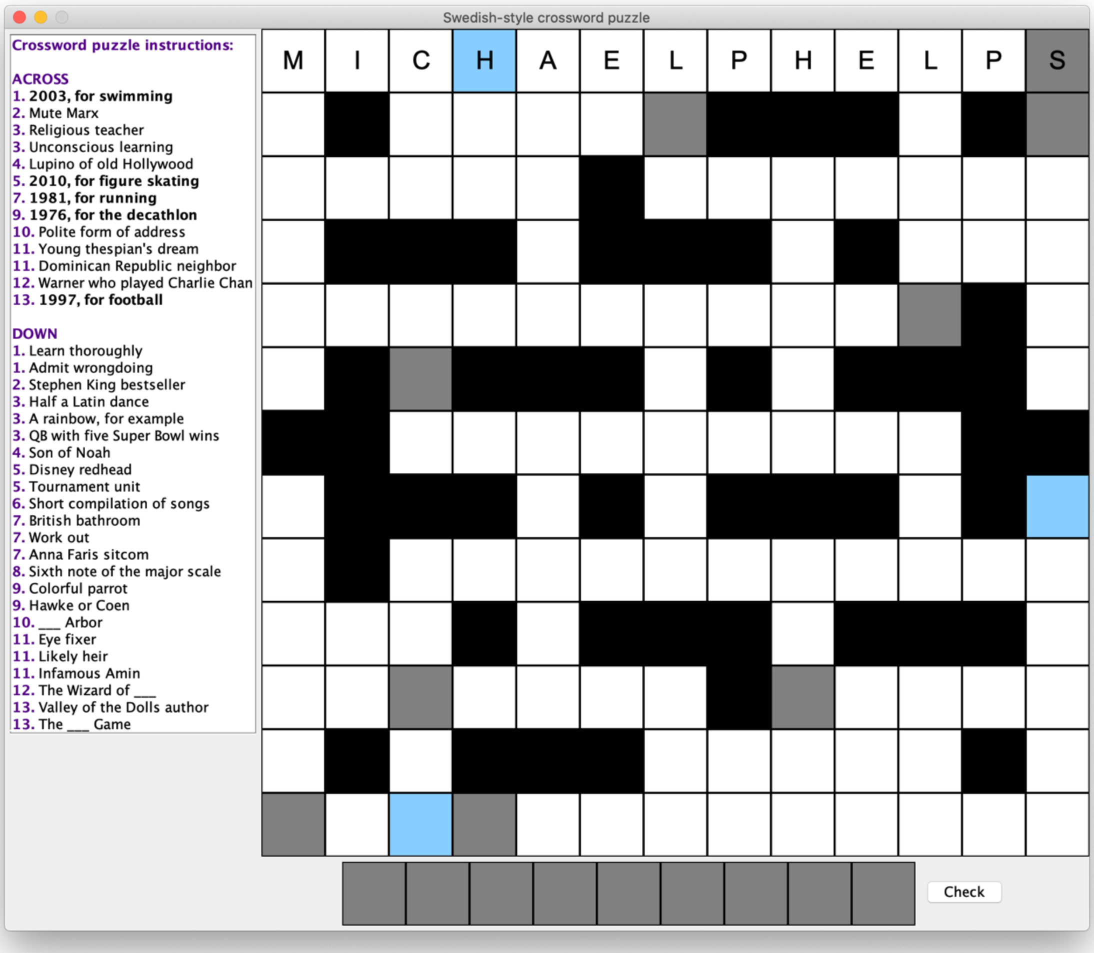
- Black squares: these are the only squares in which you can't enter any letter / character; all other squares can take 1 single letter of the standard 26-character alphabet (A-Z)
- Regular / white squares: "standard" squares that take a single character but without any special functionality or purpose
- Grey squares: you need to collect the characters at these squares to form a special word (or solution) which you can enter below the playing field; note that entering the character in a grey square on the playing field does not lead to that same character being entered somewhere in the row of grey squares below the playing field
- Blue squares: these squares provide help in case you don't know which character to enter by providing 5 possible characters in the pop-up keyboard, including the correct character that is provided in the input file (see further down for an example)

Upon clicking a white, grey or blue square, a keyboard should appear from which you can select which character you would like to enter in the square you clicked (see screenshot below). After clicking the character of choice, the keyboard disappears until another square is clicked.

When you have "collected" a number of grey squares (i.e., provided the characters in those grey squares), or if you're simply feeling lucky, you can try to fill out the row of grey squares below the playing field. The characters in that row should form a word or phrase. Click the 'Check' button to have the application evaluate your solution and show a pop-up that tells the player whether or not the answer is correct. You can come up with your own design of what to show the player on this occasion (no screenshot is provided in this assignment). Note that only the user-provided characters in the grey squares below the playing field are considered to be the key solution to the puzzle. There is no need (and also no way) to check whether the character in each individual square is the correct one.

In the screenshot below, you can see an example of what happens in the application when / after entering the answer to the first clue. Indeed, Michael Phelps was the 2003 winner of the James E. Sullivan Award in 2003. When clicking on a white square, a default keyboard pops up from which you can select the character to enter:



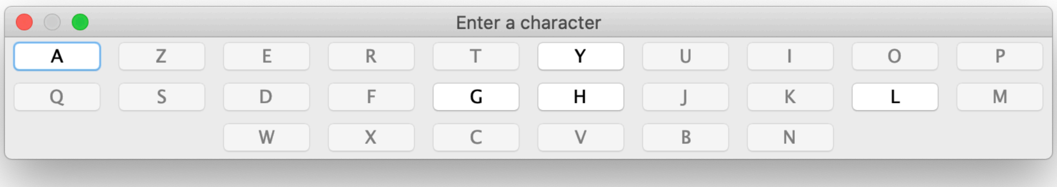


Crossword puzzle instructions:

ACROSS
1. **2003, for swimming**
2. Mute Marx
3. Religious teacher
3. Unconscious learning
4. Lupino of old Hollywood
5. **2010, for figure skating**
7. **1981, for running**
9. **1976, for the decathlon**
10. Polite form of address
11. Young thespian's dream
11. Dominican Republic neighbor
12. Warner who played Charlie Chan
13. **1997, for football**

DOWN
1. Learn thoroughly
1. Admit wrongdoing
2. Stephen King bestseller
3. Half a Latin dance
3. A rainbow, for example
3. QB with five Super Bowl wins
4. Son of Noah
5. Disney redhead
5. Tournament unit
6. Short compilation of songs
7. British bathroom
7. Work out
7. Anna Faris sitcom
8. Sixth note of the major scale
9. Colorful parrot
9. Hawke or Coen
10. ___ Arbor
11. Eye fixer
11. Likely heir
11. Infamous Amin
12. The Wizard of ___
13. Valley of the Dolls author
13. The ___ Game

When clicking on the blue square, the following keyboard pops up offering 5 choices including the correct one (i.e., H) and 4 random characters:



As you can see in the screenshot on the previous page,

Providing input to your implementation:

At this point, the only aspect left to discuss for the puzzle is how to instruct your implementation which text file to read and what the correct solution of the puzzle is (i.e., the correct order of characters in the row of grey squares below the playing field).

To this end, there are various options available. The easiest one would be to provide this information via the program arguments, but you can also make use of a properties file. Which of these approaches to implement is entirely up to you and does not affect your score in any way. You may assume the presence (and name) of such a properties file in your implementation if you so choose. More information can be found here (for example):

https://www.jetbrains.com/help/idea/properties-files.html

https://stackoverflow.com/questions/30010833/creating-a-properties-file-in-java-and-eclipse/30010882

For this assignment and the specific puzzle shown in this document, such a properties file (typically named config.properties) only needs to contain two lines of key-value pairs:

```
filename=puzzle-1-adjusted.txt
solution=southpark
```

Important: make sure that you use relative paths to the file that needs to be read so that your application works directly when copying your Eclipse / IntelliJ project files. Hard coding the location of the file into your implementation is considered poor practice!

Key points when implementing the assignment:

The structure of your code is the most important evaluation for the project. This means you should take care to include as many of the **object-oriented concepts covered in the course** whenever applicable. That also means thinking about future extensions of your implementation and keeping an eye on possible code reuse (outside of the current assignment).

Watch out for plagiarism! Online you will find similar game implementations which you may use for inspiration, but you must write your own code! Note that many code examples you can find online are poorly written and/or poorly designed.

A visually appealing GUI will obviously count towards your final grade but is not required to pass. On the other hand, an exceptional GUI without well-developed underlying logic is not sufficient. So, to summarize: a well-designed but more limited project will lead to a higher score than a poorly designed but very graphically appealing interface since we want you to focus first on the logic! We realize that design patterns such as Model-View-Controller are not part of the course material, and you are **not** meant to study this material for the project; you are however required to think about properly structuring your project code.

Finally, aim for a well-designed / well-structured project with cleanly written code, **organized in different packages**. As stated before, clearly state in the project if you were unable to provide certain requested aims of the project and provide some information (what did you try and where did it go wrong).

Questions can be asked on the Discussion Board in Toledo.

Report Guidelines

You should prepare a short report of **maximum 4 pages** detailing important components of your project. It should include the following information:

- Write a short description of every class, indicating what functionality is included in each class.
- Describe the relationship between your classes (this may be text and/or a simple diagram, but not a UML diagram), for example, inheritance relationships or method calls from one class to another.
- If you are unable to implement a certain part of the game, we encourage you to make a sensible decision, be upfront about this decision (i.e., explain it in your report) and explain what the main difficulty was. Additionally, if you encounter implementation difficulties or time constraints, it's better to fully and correctly implement a subset of the functionality rather than to implement small parts of each of the targeted program parts.
- Discuss what you think are the **strengths and weaknesses in your project** but focus on code design when doing so and not on how the application looks. Describe any difficulties you faced while working on the project.