ICITST 2006

International Conference for Internet Technology and Secured Transactions
September 11–13, 2006, London, United Kingdom

Implementing a graphical password scheme that uses nested grids

Anastasios Alexiadis, Konstantinos Chalkias and George Stephanides
University of Macedonia, Greece

Abstract: An alternative to textual password approach is the DAS ('Draw-a-Secret') scheme proposed by Jermyn et al. (1999). In this scheme the user draws a design on a display grid, which is used as the password. By making a survey on a sample of children with different age and technical knowledge, we found out that this kind of users tends to draw simple lines and shapes on specific areas in the grid. Motivating by this event, we tried to use a different construction that uses nested grids. To test this approach we implemented a proof-of-concept application, designed as a programming library, where users are encouraged to use different pre-defined grid templates or create their own one according to their needs. The results show that this approach is suitable for less technical inclined people because it helps them create more complex password representations.

Keywords: Graphical passwords, Draw-a-Secret, nested grids, grid templates, user study

1 INTRODUCTION

In commonplace alpha-numeric password schemes, users tend to choose passwords with predictable characteristics, related to how easy they are to remember. Unfortunately, since memorable passwords typically exhibit patterns, they fall into a small subset of the full password space. Various psychological studies [9] show that people can recall concrete words much better and faster than abstract words. The latter means that passwords with no meaning, such as 'sk$tr@Lm4', are even less likely to be recalled than concrete words and we expect that common users will not choose such passwords. Another case study contacted by Klein [10] resulted in 25% of 14000 passwords being cracked using a dictionary of only 3 × 106 words. This suggests that textual password-based authentication systems are susceptible to automated attacks.

To overcome the vulnerabilities of textual passwords, a lot of alternative techniques have been proposed such as visual or graphical login, biometric systems and fingerprint or voice recognition. Some of these systems are more accurate and secure, but on the other hand, they are too expensive and can only be used for specific purposes. However, visual and graphical schemes do not require specialised hardware and can be easily implemented on personal computers (PCs) and personal digital assistant devices (PDAs).

The remarkable capability of people to remember pictures motivated password schemes requiring recall or creation of a picture. These schemes can be divided into three categories at least. The first category includes schemes where a user is required to click within some regions in a picture [1][5][14][16]. The second category includes schemes where a user needs to choose a sequence of pre-defined images [3][5][6][8][13]. The third category includes schemes where a user needs to draw a design on a display grid [1][2][5][7][12][15]. In this paper we focus on the third category and more specifically we make an extension to the DAS password scheme ('Draw-a-Secret') proposed by Jermyn et al. (1999).

In the simple DAS scheme the user draws a design on a display grid, which is used as the password. A user can draw lines, symbols or even geometrical shapes. Strokes can start anywhere inside the grid and go in any direction, but must occur in the same sequence as in the registration phase. The drawing is then mapped to a sequence of coordinate pairs by listing the coordinates of the cells through which the drawing passes. In order to authenticate a user or to construct a key for symmetric encryption, the hash product (SHA-1) of the bit string that represents the sequence of coordinates (along with the pen-up characters) is used. In this way, two distinct drawings are transformed, with high probability, into two distinct hash values.

Jermyn et al. [7] have computed the full password space for the DAS scheme on a 5 × 5 grid by assuming that all passwords of total length greater than some fixed value have probability zero. According to them, the number of passwords (Lmax) of length 12 is 258 and exceeds the number of textual passwords of 8 characters or less constructed from the printable ASCII codes (≈253). Theoretically, the full password space is big enough for a user that chooses a password of twelve isolated dots or something similar, but in practice the memorable password space is much smaller, because users tend to draw lines or shapes and not isolated dots (the marked cells are usually adjacent).

There have already been proposed schemes with slight modifications of the simple DAS technique that increase the full password space. P.C van Oorschot and J.Thorpe suggested that increasing the grid size to increase the password space does not provide enough security pay-back as increasing other parameters, such as stroke-count or password length. In their analysis they assumed that when the grid increases the password length remains the same, but in practice this is not the fact. Usually, the size (in pixels) of the drawing lines remains the same before and after the increase of the grid. Under their assumption, they proposed a scheme that uses grid selection, in which a user selects a small drawing grid from an initial large, fine-grained grid. Their idea is similar to that discussed by Birget et al. (2003), except that they are zooming in on a grid to draw in, not a picture to click a point within. Another approach to improve the performance of the DAS scheme is by using a multi-grid discretization [1]. In this scheme, three grids are used in order for a point to be at safe distance from the edges from at least one of the three grids. This approach has good results when strokes pass through the corners of the cells.

In this paper we try to use a different construction of a more complex DAS scheme that uses nested grids. Unlike previous approaches, in the proposed scheme the cells in the final grid are not equal in size. We concluded in this construction by making a survey on a sample of children with different age and technical knowledge. The results show that this kind of users tends to draw simple lines and geometrical shapes on specific areas in the grid, usually in the middle or in the corners of the grid. Motivating by this event, we tried to implement a programming library that includes classes and methods needed to construct a complex DAS (CDAS) application. We used this application to test user choices and we found out that the CDAS approach is suitable for less technical inclined people because it helps them create more complex password representations.

The rest of the paper is organized as follows: §2 provides the details of the user studies and their results. In §3 there is a reference to the proposed complex DAS scheme. §4 specifies some of the basic implementation issues. Concluding remarks and future work are discussed in §5. Finally, the attached Appendix A contains information about the creation of nested grids, the password representation, some password samples from the survey and finally the UML diagram of the CDAS application.

2 USER STUDIES

In order to determine whether non-technical inclined people are able to recall graphical passwords, we conducted a study of user-drawn DAS graphical passwords. The purpose of this study was to answer the following questions:

What kind of passwords do users prefer?
What is the average password length?
Will users actually be able to remember their graphical passwords?
What are the most common mistakes when recalling a password?

Motivating by the results of the DAS case study, we conducted a second survey where users had to draw a secret on a complex grid. In the last case, we used a grid template with nested grids. After the completion of the second survey, we compared the results of the case studies in order to check if there was any improvement to the number of correct passwords.

2.1 Population and instructions

In both surveys we used a sample of 40 children (20 males and 20 females) that go to the primary school. The age of the students was between 6 and 11. The reason for choosing this sample is that we needed to test non-technical inclined people and people that had little experience with other authentication schemes such as text passwords.

Before the survey starts, specific instructions were provided to the students. We made a short course of about 25 minutes explaining the graphical password technique. Moreover, we presented some password examples (on the blackboard) and we analysed some of the graphical password features such as the way the cells are marked, the complexity and the difficulties in password representation and recalling when drawing-lines pass through the corners of the cells.

2.2 Methodology

Each of the surveys consisted of two phases. The first one was the registration phase, where the participants were asked to create a password in a specific memorable way by entering it for four times to increase the possibility of correct password recalling. Following a one-hour break and a short interview to obtain information about past computer experience, in the second phase, users were asked to recall their graphical passwords.

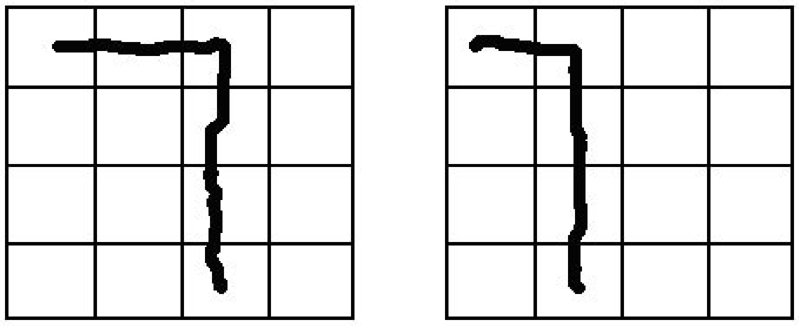
Concerning the grid size, the simple DAS survey had a 4×6 grid, while the CDAS survey had a nested grid with 21 cells. We tried to use grids with similar characteristics to provide fairness when examining the results of the user studies.

We consider a password match if it is drawn exactly as the password entered at the registration phase. Criteria for a match include the number of strokes, the order and direction of strokes and the password length.

2.3 Categorising the Results

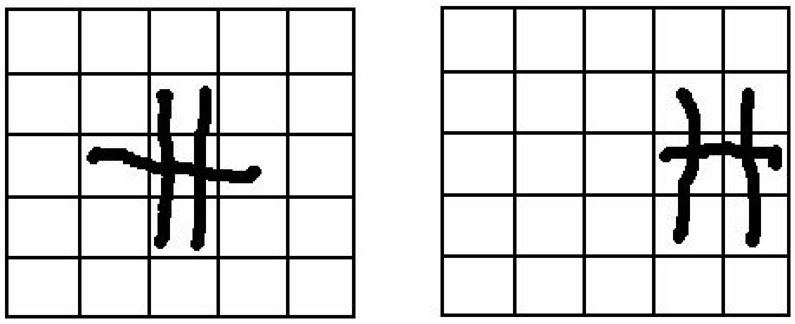
For the purpose of comparing the simple DAS scheme with the proposed CDAS scheme, the results were categorised for the following characteristics: number of strokes, password length, centering within the grid (centered, approximately centered, or not centered)[11], matched passwords, visual matched passwords (identical passwords but with errors at stroke order), shift errors (marking an adjacent cell or row of cells and not the correct one).

Figure 1. Shift Error



The number of strokes is the total number of lines and spots (or the number of pen-ups). The password length is the total number of the marked cells. A centered password means that the password is a set of marked cells adjacent to the center grid lines. An approximately centered password is a set of marked cells on either side of the center grid lines.

Figure 2. Centered and Approximately Centered Passwords



2.4 Results for the DAS scheme

This section shows the results for each of the observed characteristics in tabular form. In order to compare the results of male and female students we present the data for both genders.

Table 1. DAS Average Results for Male Students

#Strokes	#PassLength	Matched	Visual Matched	Shift Errors	Matched + Shift	V.Matched + Shift
2.9	8.1	50%	55%	25%	75%	80%

Table 2. DAS Detailed Password Complexity for Male Students

1 - 2 Strokes	3 - 4 Strokes	5 - ∞ Strokes	1 – 5 Pass. Length	6 – 9 Pass. Length	10 - ∞ Pass. Length
45%	40%	15%	25%	45%	30%

In terms of the user centering the password about the grid, 40% of the passwords were centered, an additional 30% were approximately centered and the rest 30% were not centered. Examining these results, the number of successful matches is lower than we had expected. Another survey from J. Goldberg et. al[4] concluded in the same results as ours. More specifically, only half of the students succeeded in recalling their passwords correctly and most of them (85%) used 1 to 4 strokes in their drawings. The fact that 25% of students made a shift error is very interesting as it implies that decreasing the shift error ratio could result in increasing the ratio of matched passwords.

Table 3. DAS Average Results for Female Students

#Strokes	#PassLength	Matched	Visual Matched	Shift Errors	Matched + Shift	V.Matched + Shift
3.8	9.3	45%	60%	35%	80%	95%

Table 4. DAS Detailed Password Complexity for Female Students

1 - 2 Strokes	3 - 4 Strokes	5 - ∞ Strokes	1 – 5 Pass. Length	6 – 9 Pass. Length	10 - ∞ Pass. Length
20%	60%	20%	15%	35%	50%

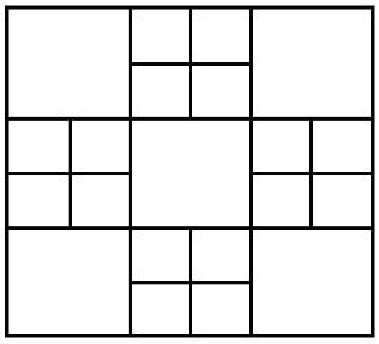
On the other hand, the survey on the female students shows that girls choose bigger passwords than boys, but they make a lot of shift and ordering errors. An interesting result is that 35% of the female students had shift errors when trying to re-enter their password. As for the password centering, the results were similar to that of the male students.

Combining the results for male and female students, we can see that the average password length is not more than 9 marked cells, while almost half of the participants recalled their password correctly. As it was referred above, the main problem of authentication failure is the shift error effect, especially for female users.

2.5 Results for the CDAS Scheme

By examining the results of the DAS survey we understood that users tend to draw on specific areas in the grid. To achieve better performance, we made another user study, but at this time using a nested grid as shown in Figure 3. In this case there are more visual base points (grid corners) on the grid that help the users to remember their previous strokes.

Figure 3. A CDAS Grid Template



As in the previous user study, we represent the results for both genders in tabular form.

Table 5. CDAS Average Results for Male Students

#Strokes	#PassLength	Matched	Visual Matched	Shift Errors	Matched + Shift	V.Matched + Shift
3.1	8.2	60%	70%	15%	75%	85%

Table 6. CDAS Detailed Password Complexity for Male Students

1 - 2 Strokes	3 - 4 Strokes	5 - ∞ Strokes	1 – 5 Pass. Length	6 – 9 Pass. Length	10 - ∞ Pass. Length
40%	40%	20%	25%	50%	25%

Unlike the DAS case study, in this complex DAS survey the password centering ratio is much smaller. More specifically, 20% of the passwords were centered, an additional 25% were approximately centered and the rest 55% were not centered. It seems that the existence of nested grids has decreased the ratio of centered passwords. Concerning the password complexity, the results show that the password length and the stroke number have been slightly increased. Moreover, the results for the matched password ratio were very optimistic as this ratio has been increased for about 15%, while on the same time there were not as much shift errors as in the simple DAS survey.

Table 7. CDAS Average Results for Female Students

#Strokes	#PassLength	Matched	Visual Matched	Shift Errors	Matched + Shift	V.Matched + Shift
3.6	9.6	60%	70%	20%	80%	90%

Table 8. CDAS Detailed Password Complexity for Female Students

1 - 2 Strokes	3 - 4 Strokes	5 - ∞ Strokes	1 – 5 Pass. Length	6 – 9 Pass. Length	10 - ∞ Pass. Length
25%	55%	20%	20%	35%	45%

The survey results for the female students are even more optimistic. The password centering ratio has been eliminated to 10%, the approximate centering is about 20% and the rest passwords are scattered to the whole grid. Furthermore, the shift errors are not more than 20% and the ratio of the matched passwords is 60%. As for the password complexity there was some improvement in the same way as in the 'male' survey.

3 COMPLEX DAS (CDAS)

Motivating by the results of the survey, we present a more complex scheme based on the DAS technique. We call this technique 'Complex DAS' as it uses nested grids for the construction of the final grid.

3.1 Motivation

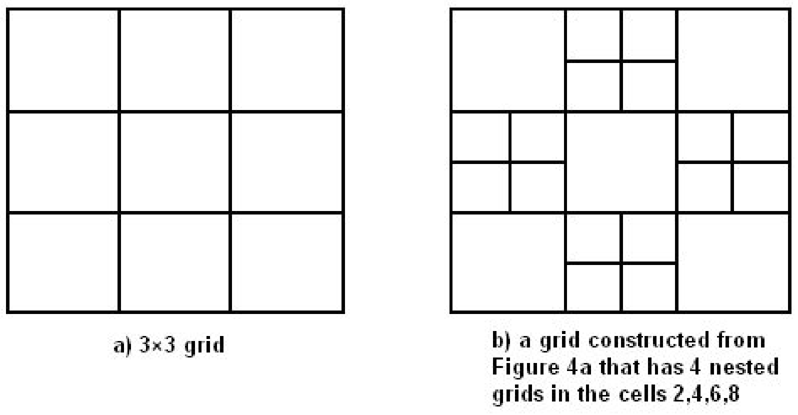
By examining the results of the DAS user study (see §2.4, §2.5) we can see that non-technical inclined people tend to draw shapes and lines on specific areas in the grid. One of the most interesting parameters is the password centering within the grid, which makes the passwords predictable for the attackers. The survey shows that the majority of users use base points on their mind and then they draw their design. This is the reason that there are a lot of centered passwords and passwords that pass from the grid boundaries (grid corners). Another important factor for password mismatch is that a big number of shift errors occur.

The results of the second survey suggest that, when there are a lot of base points on the drawing grid, the match ratio is increased. Motivating by these results we tried to construct a different much more complex scheme that uses nested grids. To improve the results we also suggest the use of grid templates, where a user would be able to select his drawing grid from a list of pre-defined grid templates.

3.2 CDAS Construction

Unlike the simple DAS technique where the cells are identical in size, in the CDAS scheme the final grid is a multi-grid consisted of internal grids. We present an example of this construction in Figure 4. In this case, the final grid could contain a lot of grid layers, but in practice two, three or four layers are enough to provide usability.

Figure 4. Creation of a Nested Grid

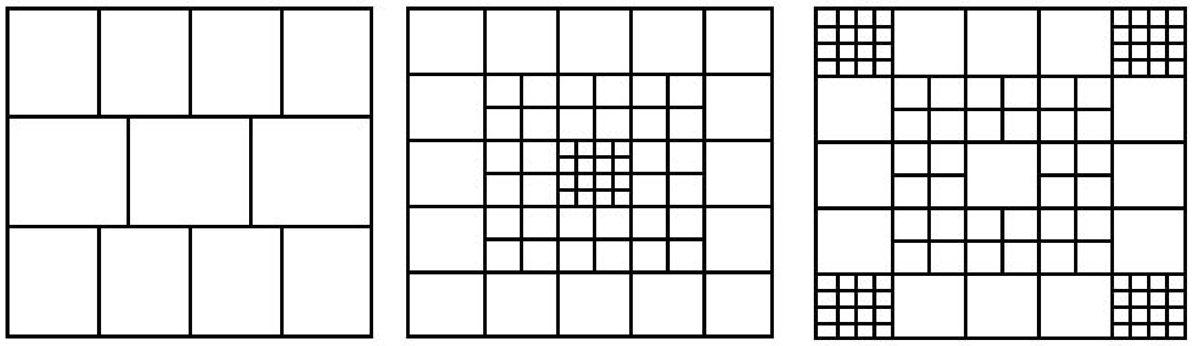

a) 3×3 grid
b) a grid constructed from Figure 4a that has 4 nested grids in the cells 2,4,6,8

As it can been seen, in the nested grid example there are a lot of visual base points, so an attacker will have problems to determine the area in where the password is centered (if it is). The password representation remains the same as the simple DAS scheme with a slight modification. For further details one can see the Appendix A.

3.3 The Use of Grid Templates

In order for a scheme to be user-friendly we suggest that the users should use different grid templates according to their needs. Every user should be able to select a grid template from a list of pre-defines templates. An application should give the ability to the users to create custom nested grids. In this way, the scheme becomes even more user-friendly, while the variety of templates could make the scheme resistant to dictionary attacks. An attacker needs to create dictionaries for different templates. Moreover, the CDAS scheme has the advantage that an attacker cannot use the "rule" that a cell has at least four neighbours, as theoretically more than four adjacent cells can exist. For example, the cell in the absolute middle of the full grid in Figure 4.b has 8 adjacent cells. Furthermore, it is understood that a multi-grid could be constructed in different ways and our system becomes a bit more resistant to shoulder surfing attacks [14].

Figure 5. Pre-defined templates



3.4 CDAS Password Space

For evaluating the security of CDAS, as well as examining how it compares to DAS, a set of recurrent functions were implemented based on the ones presented in [7]. These functions calculate the raw password space for DAS, the set of all possible passwords with either stroke-count =1 or with stroke-count≤length, were length defines the length of a password. For more information about these functions one can see [7]. For the CDAS template we used in the survey, some of the functions were altered to search the non-standard neighbors of this template.

Table 9. Size of raw password space

	DAS 4×4 (16 squares)	DAS 5×5 (25 squares)	CDAS template from survey (21 squares)	DAS 3×7 (21 squares)
Length ≤ 4 Stroke-count =1	704	1285	1781	945
Length ≤ 9 Stoke-count =1	249864	628945	2286581	341927
Length ≤ 4 Stroke-count ≤ 4	116160	581960	335340	305152
Length ≤ 4 Stroke-count ≤ 9	2.8973e+11	1.0412e+13	3.1812e+12	2.4634e+12

Table 9 compares the raw password space between the CDAS template used in the survey with common DAS templates. The results show a great increase in raw password space, even against DAS 5×5 which has 4 squares more than the CDAS template used in the comparison. When the raw password space is calculated with the stroke-count number as $1 \le x \le \text{length}$, where x is the number of stroke-counts, the results are almost the same with DAS.

By taking a close look at the results, one can observe that the raw password-space, especially when using low stroke counts, depends heavily on the number of neighbours of each square. The latter means that except from the total number of neighbours on a grid, another factor that plays an important role on the increment of the password space is the existence of cells with a large number of neighbours. As it is already referred in § 3.3, in the CDAS schemes there are cells with a big number of neighbours (e.g. the cell in the middle of the proposed CDAS template has eight adjacent cells). The aforementioned explains the fact that the raw password space of the proposed CDAS template (21 cells) is greater than the one of the simple 5×5 DAS (25 cells) when low stroke counts are used.

4 SOFTWARE IMPLEMENTATION

In this section we describe the implementation of the programming library that is required to build an application that provides CDAS authentication. The UML diagram is represented on Appendix A.

4.1 Grid Representation

One of the main points for the library implementation is the representation of the nested grids. There is an abstract super class, called GridRegion that implements the Java serializable interface. The state of this class holds the ID of the region, a reference to its parent, its pixel-coordinates and its relative row and column index (inside the parent's cell). The ID of the region is an array of integers, where the first n-1 elements of the array is the ID of its parent and the n-th element is this region's index in its parent. We represent the n-th element according to the formula shown in the following equation. There are methods to return the above information, a method to setup the coordinates of the region and a method to test the equality of two regions by comparing their IDs.

$$\text{column} + (\text{row}-1)\times\text{maxColumns} \qquad (1)$$

There are three classes that extend the abstract GridRegion class, which are the GridRegionContainer class, the Square class and the Grid class. The GridRegionContainer represents a nested region in the grid. The state of this class holds an array of its children, represented as GridRegions. Moreover, it contains the number of subrows and subcolumns of this region and a Boolean variable that is defined as true if its children are instances of the GridRegionContainer. There are methods to return the children and the Boolean variables and there is also a method to create the children. This method gets for parameters the number of subrows and subcolumns and whether the children will be instances of the GridRegionContainer.

The Square class represents a cell of the final grid. Its state is an integer variable, called marked, that holds the number of times this cell has been marked by the user. There are also methods to increment and decrement the value of the 'marked' variable.

The abstract Grid class represents the complete nested grid. The state of this class holds an array of GridRegions, which are the main regions of the grid. It also holds an instance of the 'GridRegionGroup' class, which has references to every Square instance of the grid. In addition, it contains an ArrayList, which provides as a placeholder of references for the Square instances used in the password sequence. There are getter methods for the main regions, the SquareGroup and the password sequence (it is returned as a Square array). The other public implemented methods are: the mark(x,y) method, the penup() method, the clearGrid() method and the getSquare(x,y) method. The mark method marks a Square. The marked square is then added to the end of the password sequence. This method doesn't add a square if it is the same as the last marked square and no pen-up action was given in between (to avoid multiple marks during mouse-drag events). As for the penUp method, it issues a pen-up action and adds it at the password sequence. It does not issue a pen up action if the last action given was already a pen up. The clearGrid() method is used to clear the password sequence. As for the getSquare(x,y) method, it is used to return the Square instance that its area contains the '(x,y)' pixel coordinates. As this is an abstract class it also includes an abstract method createGrid() that defines and creates the actual grid. Moreover this class includes a protected method drawGrid(graphics2D, width, height) that sets the region sizes of all the subregions and it draws the grid. Furthermore, there are three private methods: the setRegionSize(parameters), the drawSubRegions(parameters) and the drawRegion(parameters). The first method is a recursive function that sets the pixel coordinates of the regions. The second one is a recursive function that is used to draw all the nested regions. The third method draws a grid layer.

There are also two other minor classes, the PenUpAction and the GridRegionGroup class. The first one is a subclass of Square representing a pen-up action, while the second one is used to hold a list of Square instances.

4.2 Templates

The Template class is a static class that serves as a placeholder of grid creation functions. The two main functions are the standardTemplate(rows, columns) and the newCustomTemplate(rows[], columns[]). Concerning the first function, it is used to create a standard DAS non-nested Grid instance. It works in the following way: It defines and initialises an anonymous inner class that extends the abstract Grid class and implements the createGrid method in order to create a standard Grid. The newCustomTemplate function works like the previous one, but it is much more complicated and calls a recursive function so as to create a custom nested Grid. Some other functions are also included to create a variety of pre-defined templates.

4.3 Complex-DAS user interface

There is a package-visible class called DrawPanel. This class extends the Java Swing JPanel that graphically displays a grid where the user can draw on by using the drawGrid method of a pre-specified Grid instance. We need to catch 3 of the mouse events: mouse-drag, mouse-click and mouse-release. When a mouse-click or mouse-drag event occurs, we draw on the grid according to the user's mouse motion. Moreover, we call the mark method of the Grid instance with the pixel coordinates of the current mouse pointer's position. When a mouse-release event occurs, the penUp method of the Grid instance is called.

The user interface includes a TemplatePanel that is used for displaying a list of pre-defined grid. Then, the client can choose one grid-template from this list to use. This class uses the DrawPanel class to display each of the pre-defined grids, but with the mouse-listeners disabled.

The main class of the CDAS implementation is the CDASPanel that extends the Java Swing JPanel and makes use of the DrawPanel for drawing purposes. There are two modes of operation: password creation and password comparison. In the first case, the program sets a Grid instance that could have been obtained by the template panel. After drawing the password, the user is able to store the grid with the hash product of his password. In the second case, the program loads the user's pre-defined grid together with the hash product of the user's password. Then, the application allows the user to draw a design. The hash product of this drawing is created and the equality of the two hash products is checked.

4.4 Password Representation and Hashing

A static class CDASUtils includes a function called hash(Square[]) that returns the SHA-1 product of the user's password. This is done by first creating a string representation of the Square instance array. The format of this representation is the ID elements of each Square, separated by commas. To distinguish the Squares, a dash symbol is used. The pen-up action is represented as 'PU'. After the string creation, the SHA-1 product is computed by the hash function.

5 CONCLUSION AND FUTURE WORK

The user studies presented support that users choose graphical passwords with predictable characteristics. In the complex DAS scheme the multi-grid gives the opportunity to the users to center their passwords in different areas on the grid. The latter makes this scheme more resistant to dictionary attacks than the simple DAS scheme. Another remark is that the ratio of the matched passwords has been improved significantly. Moreover the survey suggests that female students tend to draw bigger passwords, but password mismatches occur much more often than the male students' passwords. Furthermore, the implementation of the Java library for CDAS authentication would be very useful for researchers to make new surveys and improve the programming code. In the future it would be interesting to conduct a survey where users will have to login periodically, so as to test the user's ability to remember long-lasting passwords. Another future improvement would be to use colours in the grid area. In this way, users will better highlight some areas in the grid.

REFERENCES

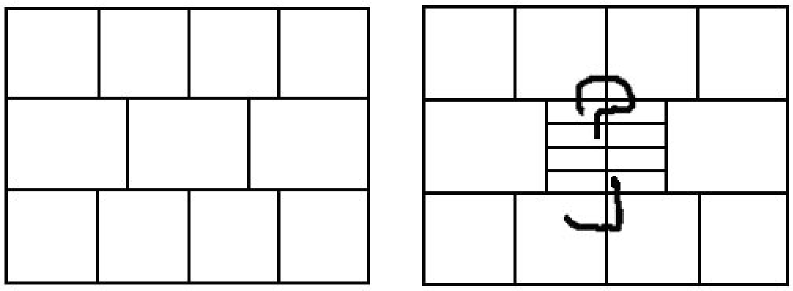
1. J.-C. Birget, D. Hong, and N.Memon. (2003), 'Robust discretization with an application to graphical passwords', Cryptology ePrint Archive, Report 2003/168; http://eprint.iacr.org (May 2006)
2. D. Davis, F. Monrose, and M.K Reiter. (2004), 'On user choice in graphical password schemes', In 13th USENIX Security Symposium.
3. R. Dhamija and A. Perrig. (2000), 'Déjà Vu: a user study using images for authentication', In 9th USENIX Security Symposium.
4. J. Goldberg, J. Hagman, and V. Sazawal. (2002), 'Doodling our way for better authentication', CHI '02 extended abstracts on Human Factors in Computer Systems.
5. W. Jansen. (2003), 'Authenticating users on handheld devices', In the Canadian Information Technology Security Symposium.
6. W. Jansen, S. Gavrila, V. Korolev, R. Ayers, R. Swanstrom. (2003). 'Picture password: A visual login technique for mobile devices', NISTIR 7030,;http:// csrc.nist.gov/publications/nistir/ nistir-7030.pdf
7. I. Jermyn, A. Mayer, F. Monrose, M. Reiter, and A. Rubin. (1999), 'The design and analysis of graphical passwords', In 8t h USENIX Security Symposium.
8. Y.-G. Kim, T. Kwon, (2004), 'An Authentication Scheme Based Upon Face Recognition for the Mobile Environment, CIS 2004, 274-279
9. E. A. Kirkpatrick. (1894), 'An experimental study of memory', Psychological Review,1:602 - 609.
10. D. Klein. (1990), 'Foiling the Cracker: a survey of, and improvements to, password security', In 2nd USENIX Security Workshop. 5-14.
11. D. Nali and J. Thorpe. (2004), 'Analysing user choice in graphical passwords', Tech. Report TR-04-01, School of Computer Science, Carleton University, Canada.
12. P.C. van Oorschot, J. Thorpe. (2005), 'On the Security of Graphical Password Schemes', Technical Report TR-05-11. Integration and extension of USENIX Security 2004 and ACSAC 2004 papers.
13. A. Perrig and D. Song. (1999), 'Hash Visualization: a new technique to improve real-world seurity', In International Workshop on Cryptographic Techniques and E-Commerce, pages 131-138.
14. L. Sobrado, J.-C. Birget. (2002) 'Graphical passwords', The Rutgers Scholar, vol.4; http://RutgersScholar.rutgers.edu/volume04/contents.htm (May 2006)
15. J. Thorpe and P. van Oorschot.(2004), 'Graphical dictionaries and the memorable space of graphical passwords', In 13th USENIX Security Symopsium.
16. S. Wiedenbeck, J. Waters, J. Birget, A. Brodskyi, and N. Memon. (2005). 'Passpoints: design and longitudinal evaluation of a graphical password system', International J. of Human-Computer Studies (Special Issue on HCI Research in Privacy and Security) 63, 102-127.

APPENDIX A

Nested Grid template creation

The template 'extendedBricks' shown in Figure 6.b is created by calling the newCustomTemplate(rows[], columns[]) in the Templates class and passing the following arguments: rows[] = {3, 1, 1, 1, 1, 1, 1, 1, 1, 4, 1, 1, 1, 1, 1}, columns[] = {1, 4, 3, 4, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1}. The function creates a Grid instance with the following properties: Firstly, it creates a main 3×1 grid and then it passes from all these regions serially (left to right first then top to bottom) and for each of the regions creates the second-layer of sub-grids. So, for the first region it creates a 1×4 grid, for the second a 1×3 grid and for the third a 1×4 grid. Up to this point, the grid is the same with the 'bricks' template shown in Figure 6.a. Now, if we take all the regions of the second-layer serially, we create for each of the first five a 1×1 sub-grid (so we actually leave these regions as they are) and for the sixth one we create a 4×2 sub-grid. Lastly we define the remaining five regions of the second-layer as 1×1. The Grid instance returned from the function with these parameters represents a three-layer grid.

Figure 6. Implementation of a nested grid



String representation example

This is the password made from the drawing in Figure 6.b

Password: 2,2,1-1,2,1-1,3,1-2,2,2-2,2,1-2,2,3-PU-3,2,1-3,3,1-2,2,8-PU

SHA-1 product: 1A:8F:6B:E4:05:3E:80:BD:2B:8F:50:48:ED:18:C0:90:F1:86:B2:26

The template used is a three-level nested grid. Each square's id is separated from the next by a dash and each square's id elements are written with commas in between. PU, as already noted, represents the pen-up action. The IDs represent a square according to the design of Grid template. In order for someone to understand the representation of a square, firstly he has to understand the design of the template used. The password in Figure 6.b represents a design that starts from square {2,2,1} and then passes serially from {1,2,1}, {1,3,1}, {2,2,2}, returns to {2,2,1} and ends in {2,2,3} where a pen-up action is issued. Then we have a second design, starting from {3,2,1}, passing from {3,3,1} and ending in {2,2,8} where a pen-up is issued again and the password ends.

User Password samples

Figure 7 presents some password samples from the case study.

Figure 7. Random DAS and CDAS Passwords

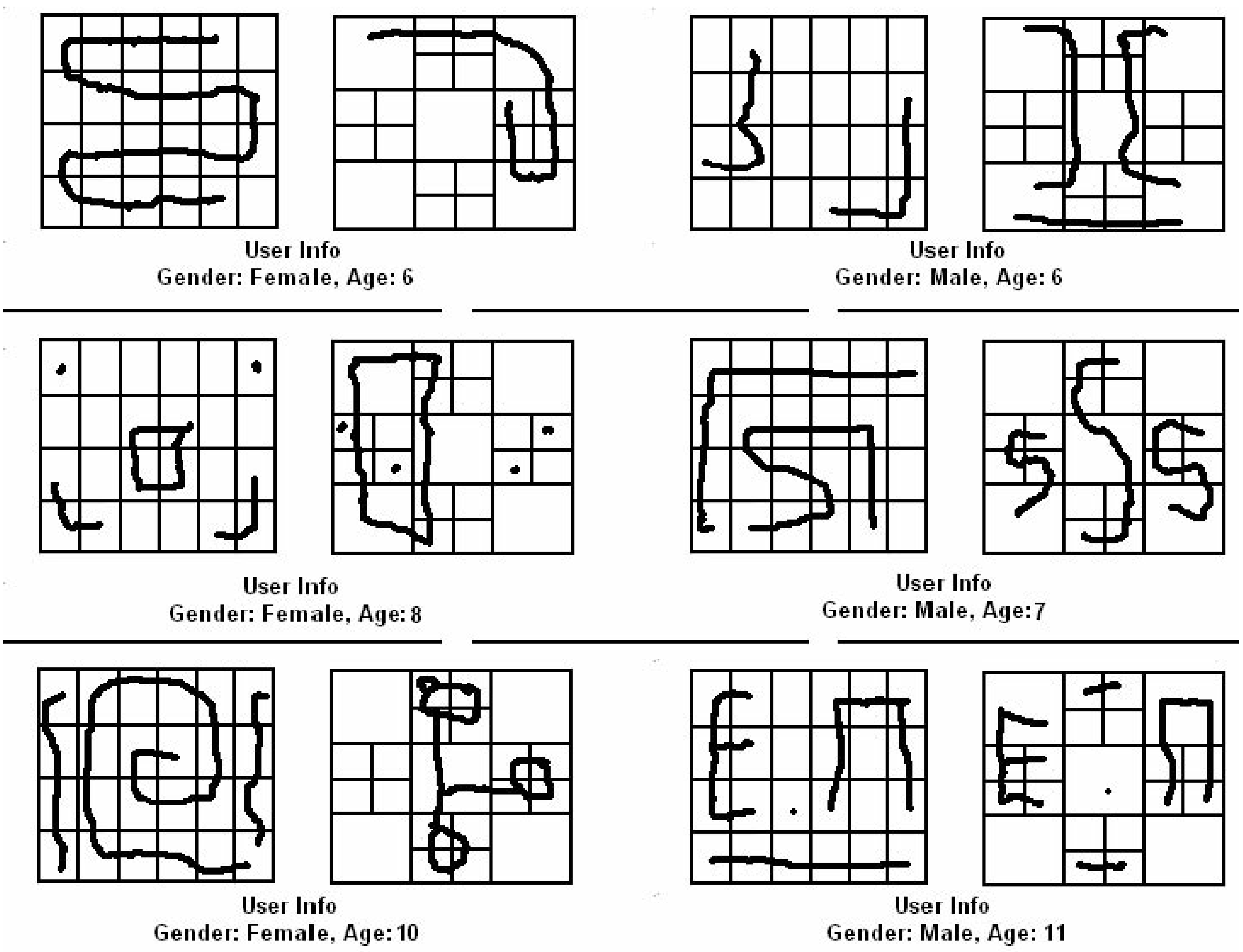
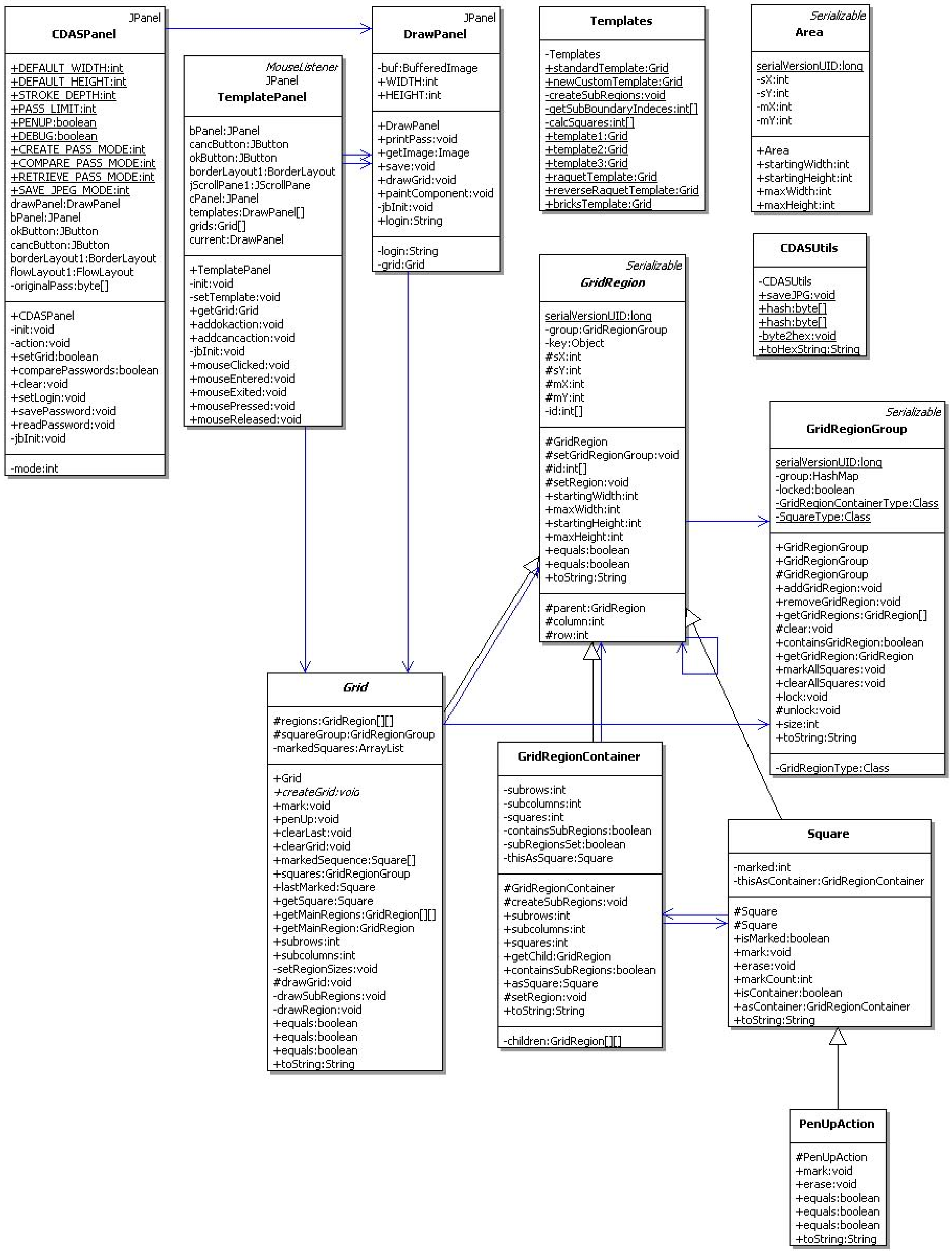


Figure 7. CDAS Project UML Diagram



JPanel
CDASPanel
+DEFAULT_WIDTH:int
+DEFAULT_HEIGHT:int
+STROKE_DEPTH:int
+PASS_LIMIT:int
+PENUP:boolean
+DEBUG:boolean
+CREATE_PASS_MODE:int
+COMPARE_PASS_MODE:int
+RETRIEVE_PASS_MODE:int
+SAVE_JPEG_MODE:int
drawPanel:DrawPanel
bPanel:JPanel
okButton:JButton
cancButton:JButton
borderLayout1:BorderLayout
flowLayout1:FlowLayout
-originalPass:byte[]
+CDASPanel
-init:void
-action:void
+setGrid:boolean
+comparePasswords:boolean
+clear:void
+setLogin:void
+savePassword:void
+readPassword:void
-jbInit:void
-mode:int
MouseListener
JPanel
TemplatePanel
bPanel:JPanel
cancButton:JButton
okButton:JButton
borderLayout1:BorderLayout
jScrollPane1:JScrollPane
cPanel:JPanel
templates:DrawPanel[]
grids:Grid[]
current:DrawPanel
+TemplatePanel
-init:void
-setTemplate:void
+getGrid:Grid
+addokaction:void
+addcancaction:void
-jbInit:void
+mouseClicked:void
+mouseEntered:void
+mouseExited:void
+mousePressed:void
+mouseReleased:void
JPanel
DrawPanel
-buf:BufferedImage
+WIDTH:int
+HEIGHT:int
+DrawPanel
+printPass:void
+getImage:Image
+save:void
+drawGrid:void
+paintComponent:void
-jbInit:void
+login:String
-login:String
-grid:Grid
Templates
-Templates
+standardTemplate:Grid
+newCustomTemplate:Grid
-createSubRegions:void
-getSubBoundaryIndeces:int[]
-calcSquares:int[]
+template1:Grid
+template2:Grid
+template3:Grid
+raquetTemplate:Grid
+reverseRaquetTemplate:Grid
+bricksTemplate:Grid
Serializable
Area
serialVersionUID:long
-sX:int
-sY:int
-mX:int
-mY:int
+Area
+startingWidth:int
+startingHeight:int
+maxWidth:int
+maxHeight:int
CDASUtils
-CDASUtils
+saveJPG:void
+hash:byte[]
+hash:byte[]
-byte2hex:void
+toHexString:String
Serializable
GridRegion
serialVersionUID:long
-group:GridRegionGroup
-key:Object
#sX:int
#sY:int
#mX:int
#mY:int
-id:int[]
#GridRegion
#setGridRegionGroup:void
#id:int[]
#setRegion:void
+startingWidth:int
+maxWidth:int
+startingHeight:int
+maxHeight:int
+equals:boolean
+equals:boolean
+toString:String
#parent:GridRegion
#column:int
#row:int
Serializable
GridRegionGroup
serialVersionUID:long
-group:HashMap
-locked:boolean
-GridRegionContainerType:Class
-SquareType:Class
+GridRegionGroup
+GridRegionGroup
#GridRegionGroup
+addGridRegion:void
+removeGridRegion:void
+getGridRegions:GridRegion[]
#clear:void
+containsGridRegion:boolean
+getGridRegion:GridRegion
+markAllSquares:void
+clearAllSquares:void
+lock:void
#unlock:void
+size:int
+toString:String
-GridRegionType:Class
Grid
#regions:GridRegion[][]
#squareGroup:GridRegionGroup
-markedSquares:ArrayList
+Grid
+createGrid:void
+mark:void
+penUp:void
+clearLast:void
+clearGrid:void
+markedSequence:Square[]
+squares:GridRegionGroup
+lastMarked:Square
+getSquare:Square
+getMainRegions:GridRegion[][]
+getMainRegion:GridRegion
+subrows:int
+subcolumns:int
-setRegionSizes:void
#drawGrid:void
-drawSubRegions:void
-drawRegion:void
+equals:boolean
+equals:boolean
+equals:boolean
+toString:String
GridRegionContainer
-subrows:int
-subcolumns:int
-squares:int
-containsSubRegions:boolean
-subRegionsSet:boolean
-thisAsSquare:Square
#GridRegionContainer
#createSubRegions:void
+subrows:int
+subcolumns:int
+squares:int
+getChild:GridRegion
+containsSubRegions:boolean
+asSquare:Square
#setRegion:void
+toString:String
-children:GridRegion[][]
Square
-marked:int
-thisAsContainer:GridRegionContainer
#Square
#Square
+isMarked:boolean
+mark:void
+erase:void
+markCount:int
+isContainer:boolean
+asContainer:GridRegionContainer
+toString:String
PenUpAction
#PenUpAction
+mark:void
+erase:void
+equals:boolean
+equals:boolean
+equals:boolean
+toString:String