Mini-Project 1: Steganography

Lecture

Barbara Jobstmann

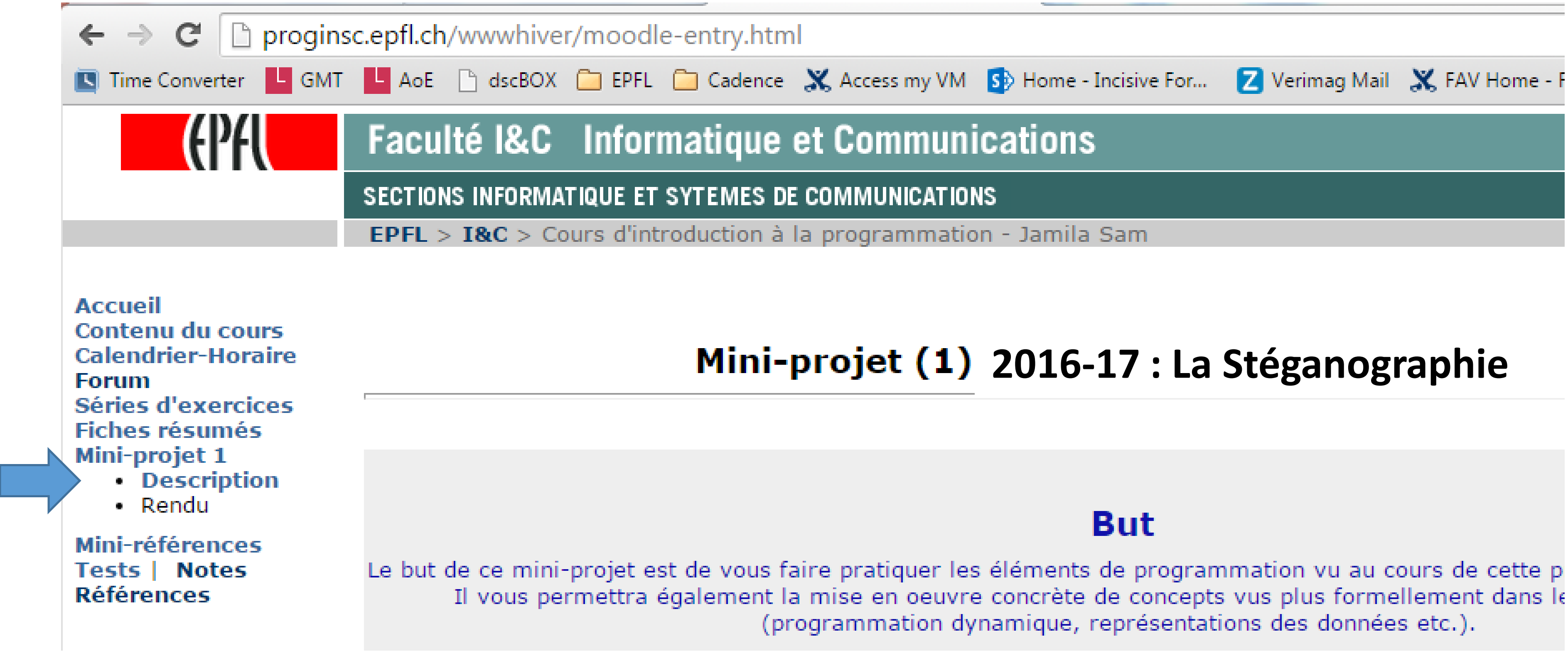
27.10.2016

Outline

- Administrative
  - Information/Starting point
  - Submission and Groups
  - Submission Server and Tokens
- Project
  - Goal
  - Overview
  - Provided Code
  - Project Details:
    - Representation of Images
    - Part 1: Concealing an image
    - Part 2: Concealing a text
    - Part 3: Spiral encoding

Information about the Project

- Detailed project description and provided material: under “Project 1” ⟶ “Description” at http://proginsc.epfl.ch/wwwhiver/moodle-entry.html



Faculté I&C Informatique et Communications

SECTIONS INFORMATIQUE ET SYTEMES DE COMMUNICATIONS

EPFL > I&C > Cours d'introduction à la programmation - Jamila Sam

Accueil
Contenu du cours
Calendrier-Horaire
Forum
Séries d'exercices
Fiches résumés
Mini-projet 1
- Description
- Rendu

Mini-références
Tests | Notes
Références

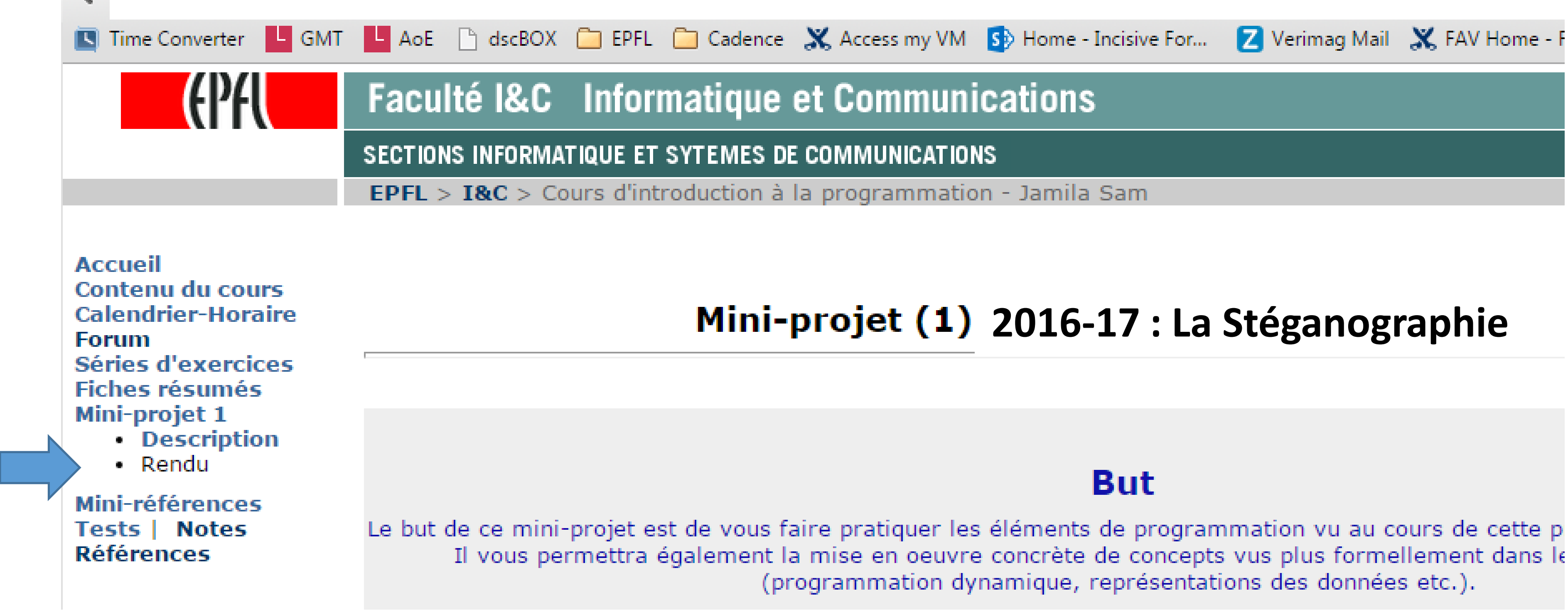
Mini-projet (1) 2016-17 : La Stéganographie

But

Le but de ce mini-projet est de vous faire pratiquer les éléments de programmation vu au cours de cette p
Il vous permettra également la mise en oeuvre concrète de concepts vus plus formellement dans le
(programmation dynamique, représentations des données etc.).

Submission

- **Deadline: Nov 14th, 1pm**
- Groups of (at most) 2 students
- Submission: under "Project 1" → "Rendu" at http://proginsc.epfl.ch/wwwhiver/moodle-entry.html



Faculté I&C Informatique et Communications

SECTIONS INFORMATIQUE ET SYTEMES DE COMMUNICATIONS

EPFL > I&C > Cours d'introduction à la programmation - Jamila Sam

Mini-projet (1) 2016-17 : La Stéganographie

But

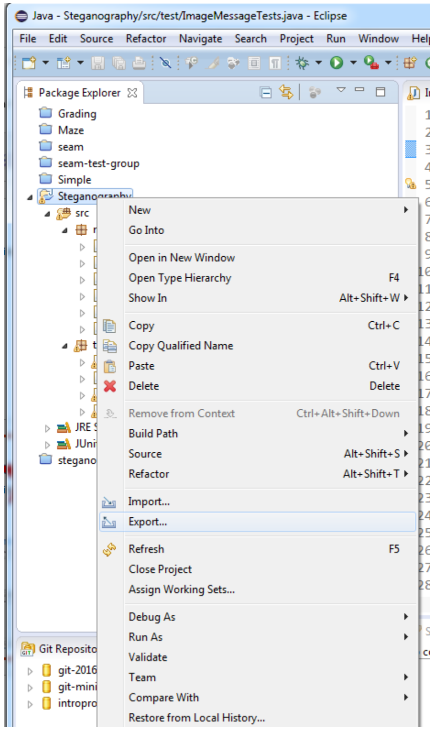
Le but de ce mini-projet est de vous faire pratiquer les éléments de programmation vu au cours de cette p
Il vous permettra également la mise en oeuvre concrète de concepts vus plus formellement dans le
(programmation dynamique, représentations des données etc.).

Submission Content

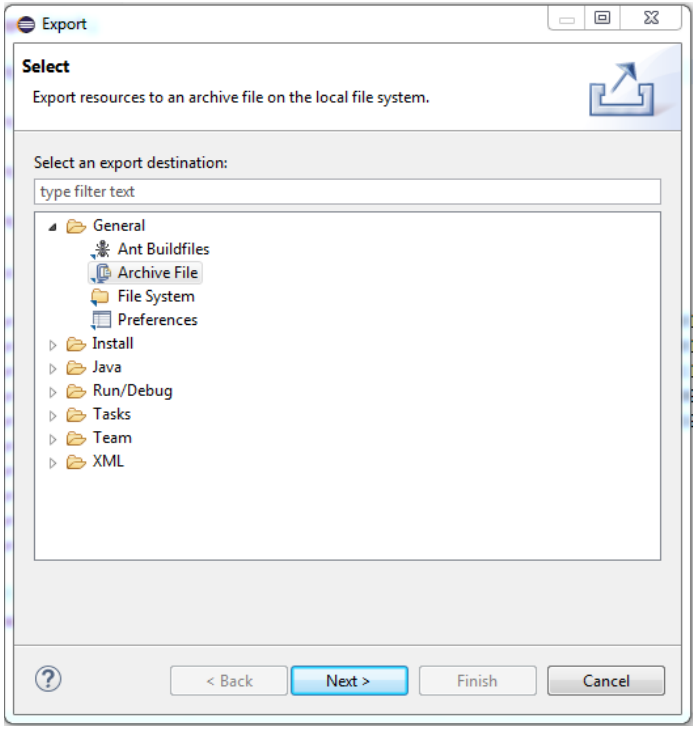
- Eclipse Archive file (zip-file < 20kB) that includes
  - ImageMessage.java
  - TextMessage.java
  - Steganography.java

Submission Content

- Eclipse Archive file (zip-file < 20kB) that includes
  - ImageMessage.java
  - TextMessage.java
  - Steganography.java



Java - Steganography/src/test/ImageMessageTests.java - Eclipse

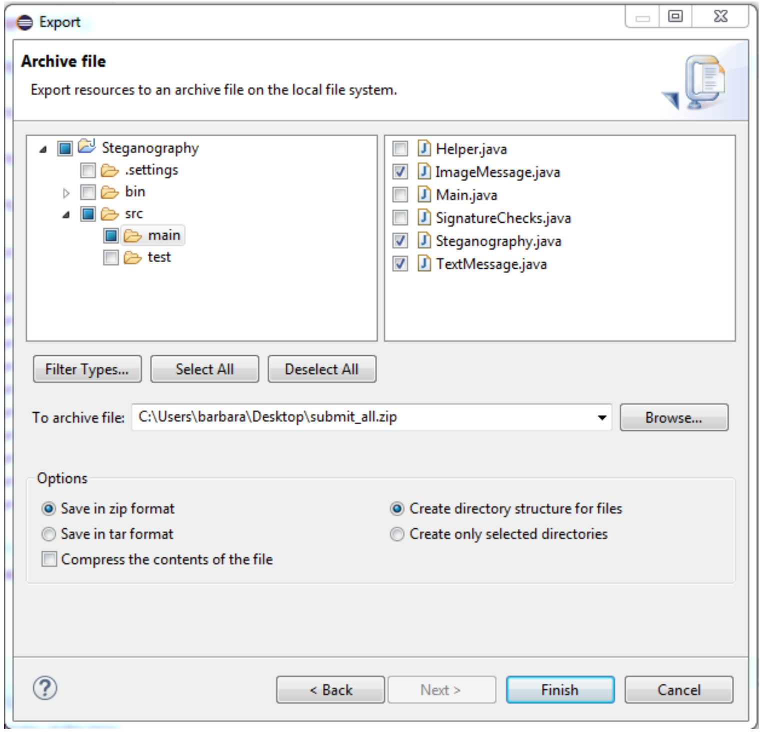


Export

Select

Export resources to an archive file on the local file system.

Select an export destination:



Export

Archive file

Export resources to an archive file on the local file system.

- ▣ Steganography
  - □ .settings
  - □ bin
  - ▣ src
    - ▣ main
    - □ test

- □ Helper.java
- ☑ ImageMessage.java
- □ Main.java
- □ SignatureChecks.java
- ☑ Steganography.java
- ☑ TextMessage.java

To archive file: C:\Users\barbara\Desktop\submit_all.zip

Options

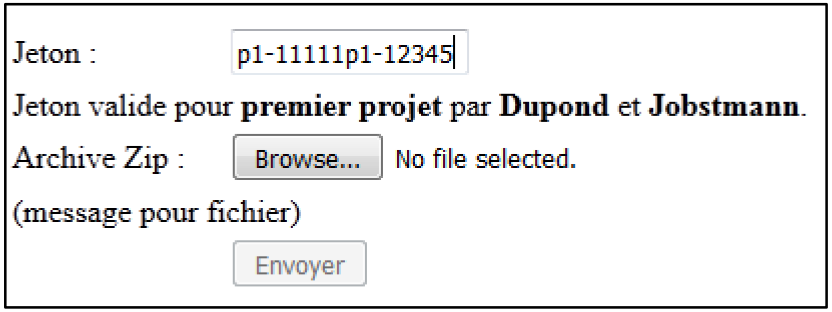
- ● Save in zip format
- ○ Save in tar format
- □ Compress the contents of the file
- ● Create directory structure for files
- ○ Create only selected directories

Submission Server

- Will open one week before the deadline:
  - From Mon, Nov 7th 9am until Fri, Nov 11th 4pm.
  - **No submissions over the weekend!**
  - Reopen on Mon, Nov 14th from 9am to 1pm (strict deadline).
- Each student will need a token (specific key) to submit.
- Tokens will be send out per email one week before the submission deadline.
- Each submission required two token: one from each group member. If you work alone, you need to use your token twice.
- You can submit a new version using the same token.
- **Hint**: submit initial (incomplete) version way before the deadline to get familiar with the submission process

Submission Server – Examples

- Example tokens: p1-11111 and p1-12345
- Example of submission with 2 students



Jeton : p1-11111p1-12345

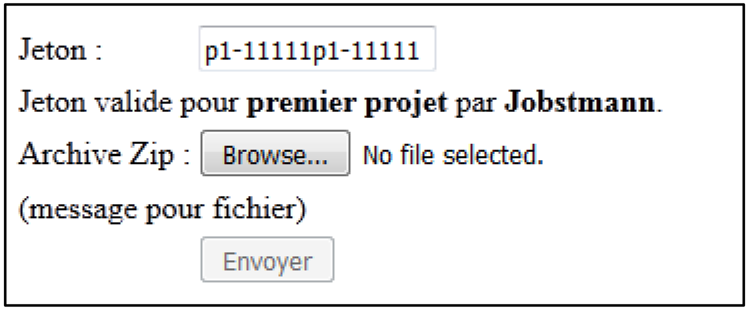
Jeton valide pour **premier projet** par **Dupond** et **Jobstmann**.

Archive Zip : Browse... No file selected.

(message pour fichier)

Envoyer

- Example of submission with 1 student



Jeton : p1-11111p1-11111

Jeton valide pour **premier projet** par **Jobstmann**.

Archive Zip : Browse... No file selected.

(message pour fichier)

Envoyer

Submission – Cheating

- The project is graded.
- The exchange of ideas between groups or with third parties is permitted and even recommended.
- **The exchange of code is strictly forbidden!**
- **Plagiarism will be controlled and will be considered cheating.**
- In case of cheating, you will receive a rating of "NA“: Art. 18 “**Fraude de l'ordonnance sur la discipline**“ https://www.admin.ch/opc/fr/classified-compilation/20041650/index.html
- Note that at anytime, you will need to be able to explain your code.

Outline

- Administrative
    - Information/Starting point
    - Submission and Groups
    - Submission Server and Tokens
- Project
    - Goal
    - Overview
    - Provided Code
    - Project Details:
        - Representation of Images
        - Part 1: Concealing an image
        - Part 2: Concealing a text
        - Part 3: Spiral encoding

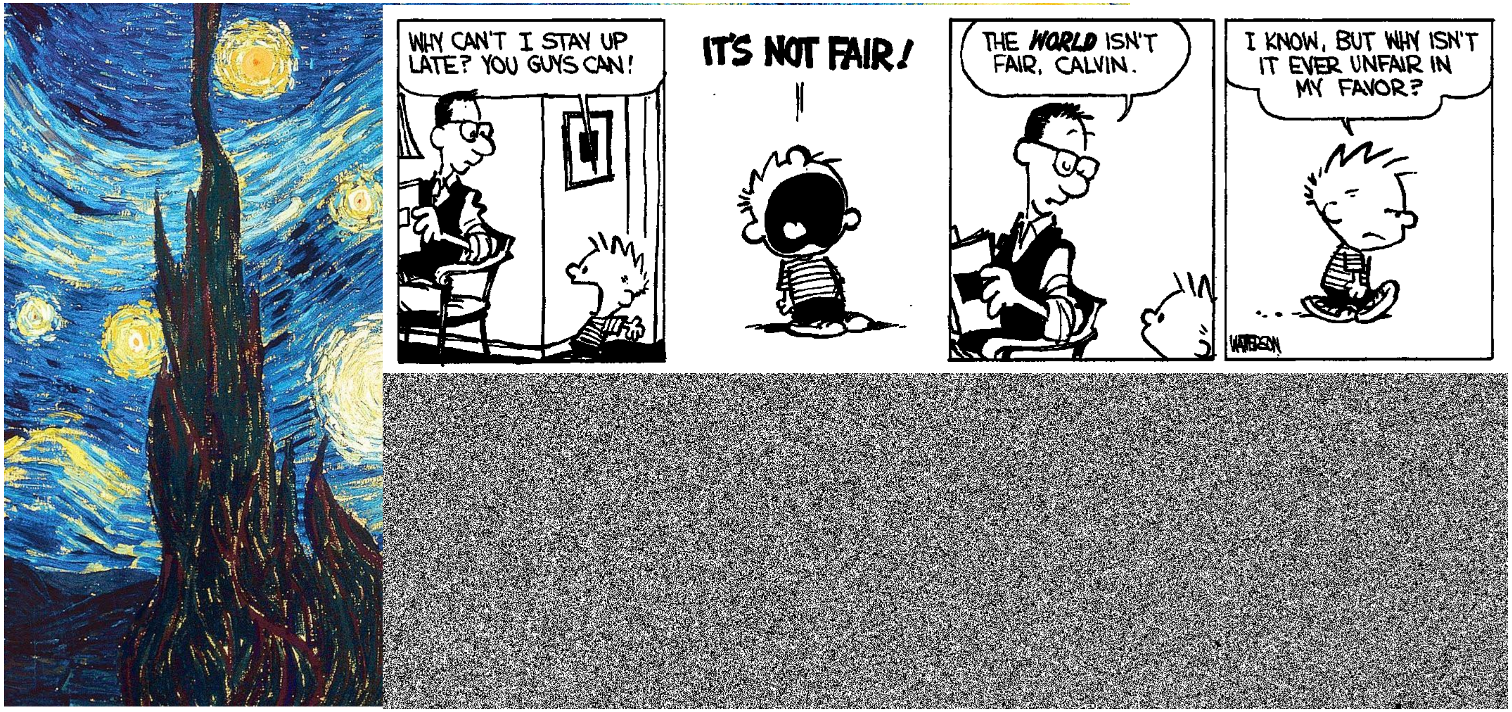
Goal

- The purpose of steganography is to hide the existence of a message from a third party.
- Advantage over cryptography alone: secret message does not attract attention

Cover Image

Goal

- The purpose of steganography is to hide the existence of a message from a third party.
- Advantage over cryptography alone: secret message does not attract attention


WHY CAN'T I STAY UP LATE? YOU GUYS CAN!
IT'S NOT FAIR!
THE WORLD ISN'T FAIR, CALVIN.
I KNOW, BUT WHY ISN'T IT EVER UNFAIR IN MY FAVOR?
WATTERSON

Cover Image **Hidden Message**

Our Approach: LSB Embedding

- Use the LSB of a pixel to store the message – visually not detectable!

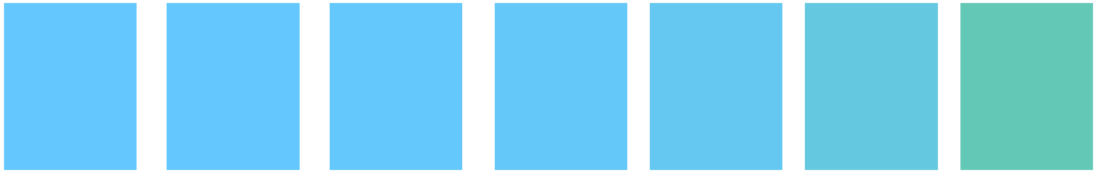
Cover Image

Cover Image + Message

WHY CAN'T I STAY UP LATE? YOU GUYS CAN!
IT'S NOT FAIR!
THE WORLD ISN'T FAIR, CALVIN.
I KNOW, BUT WHY ISN'T IT EVER UNFAIR IN MY FAVOR?
WATTERSON

Our Approach: LSB Embedding

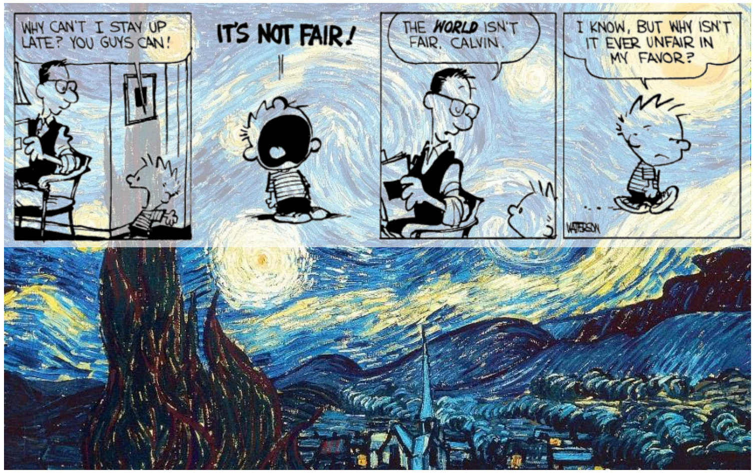
- Use the LSB of a pixel to store the message – visually not detectable!
- Each pixel has a color defined by an RGB (Red-Green-Blue) value represented by one Byte per color
- Changing LSB – changing value of blue slightly



Red	100	100	100	100	100	100	100
Green	200	200	200	200	200	200	200
Blue	255	254	252	248	240	224	182
Diff		-1	-2	-4	-8	-16	-32
LSBs		1	2	3	4	5	6

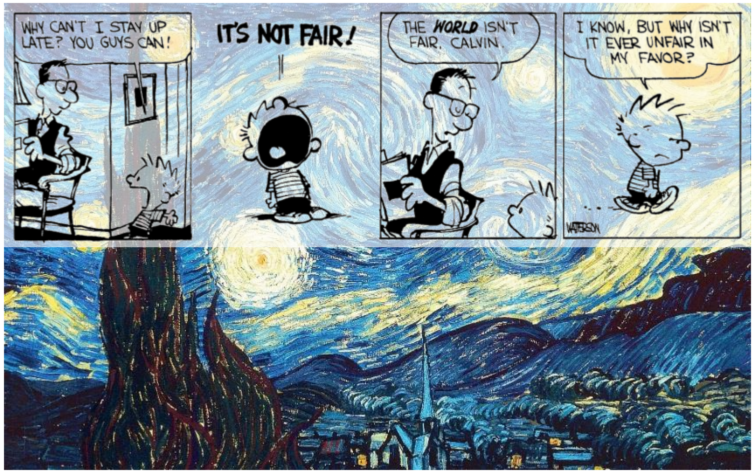
LSB Embedding Options

1. Embed black-white image linearly


WHY CAN'T I STAY UP LATE? YOU GUYS CAN!
IT'S NOT FAIR!
THE WORLD ISN'T FAIR, CALVIN.
I KNOW, BUT WHY ISN'T IT EVER UNFAIR IN MY FAVOR?
WATTERSON

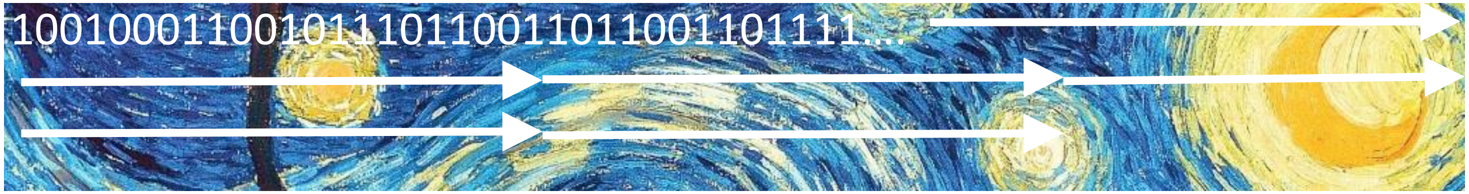
LSB Embedding Options

1. Embed black-white image linearly


WHY CAN'T I STAY UP LATE? YOU GUYS CAN!
IT'S NOT FAIR!
THE WORLD ISN'T FAIR, CALVIN.
I KNOW, BUT WHY ISN'T IT EVER UNFAIR IN MY FAVOR?
WATTERSON

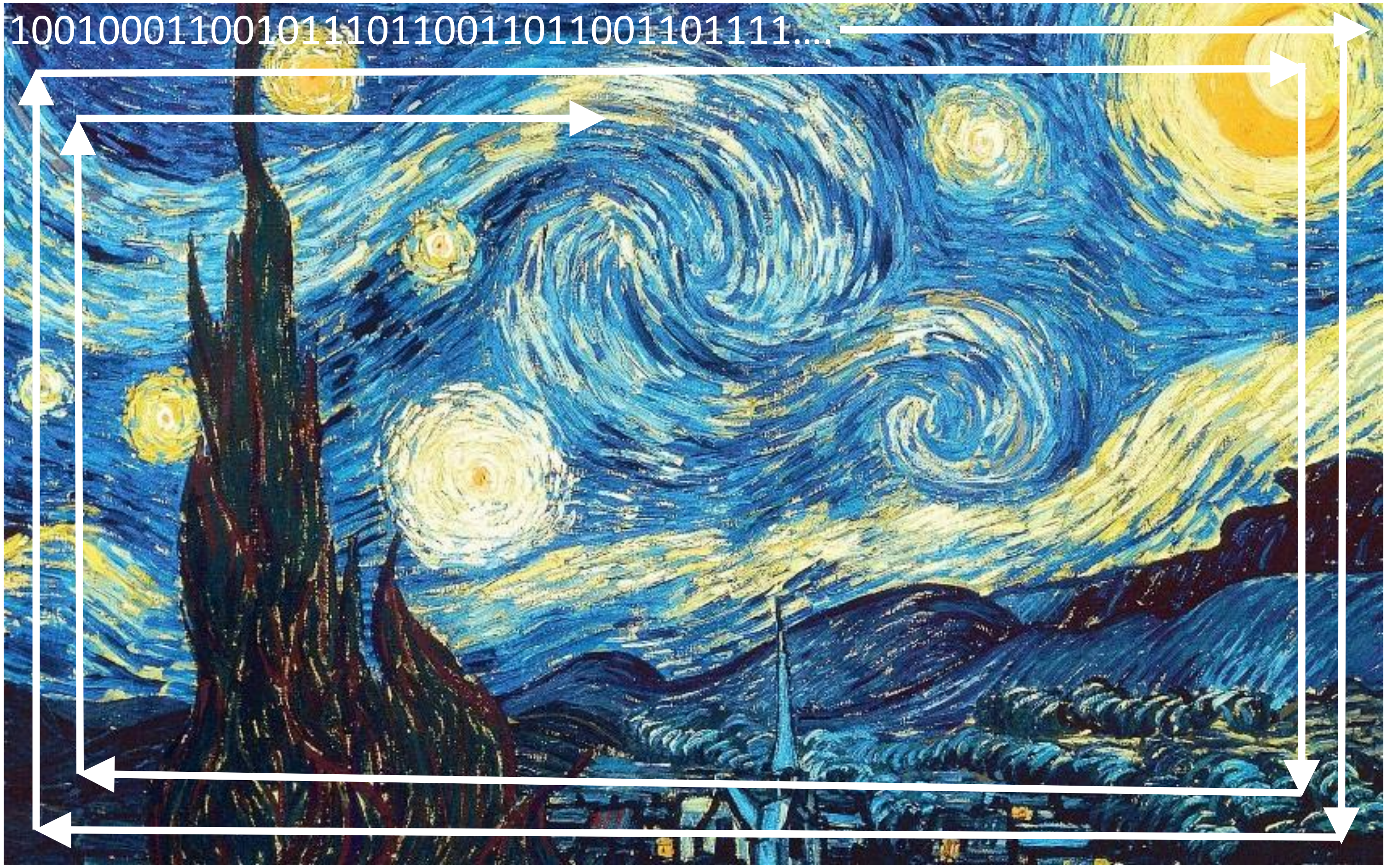
2. Embed Text linearly

Hello → ‘H’, ‘e’, ‘l’, ‘l’, ‘o’ → 72, 101, 108, 108, 111 → 1001000, 1100101, 1101100, 1101100, 1101111

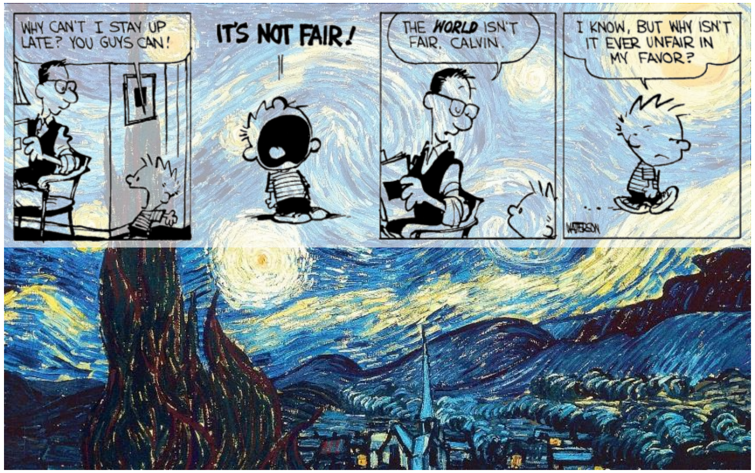


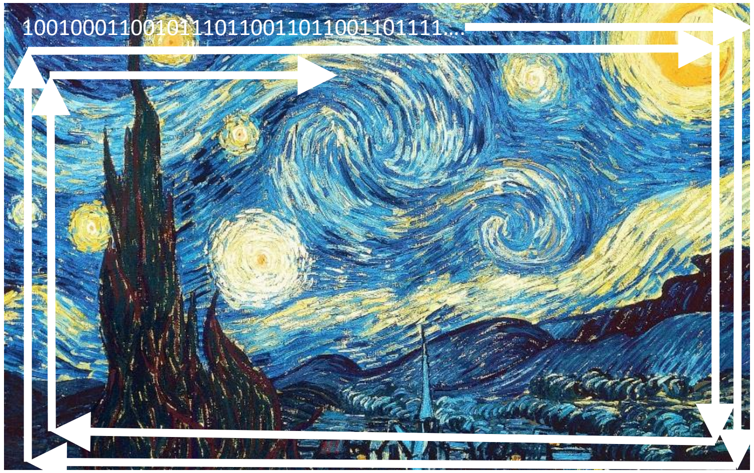
LSB Embedding Options

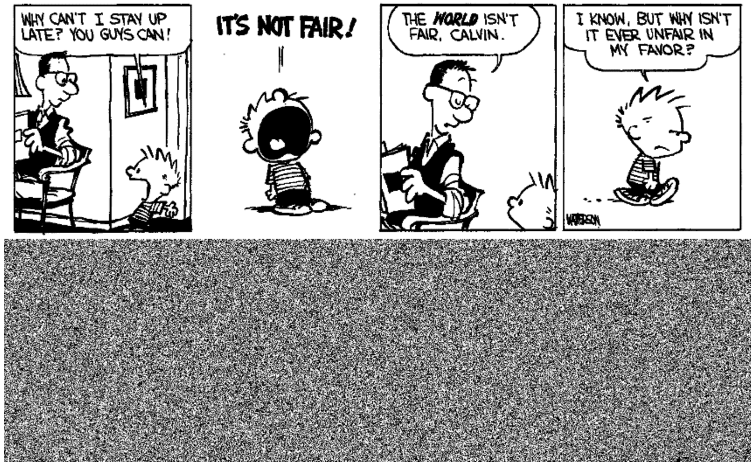
3. Spiral Embedding

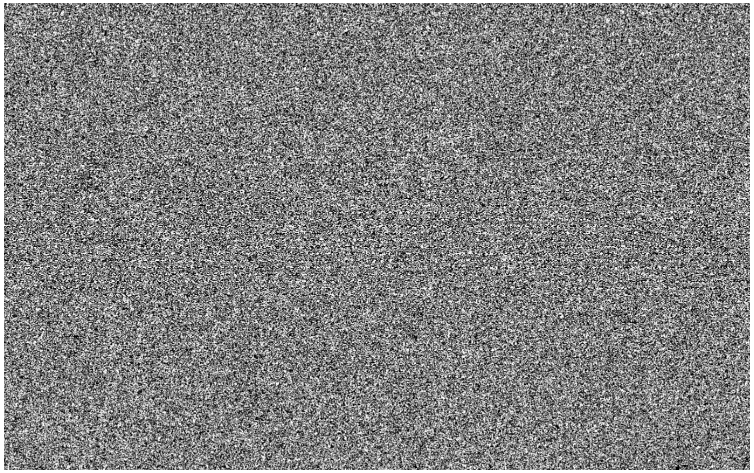


Linear versus Spiral Embedding


WHY CAN'T I STAY UP LATE? YOU GUYS CAN!
IT'S NOT FAIR!
THE WORLD ISN'T FAIR, CALVIN.
I KNOW, BUT WHY ISN'T IT EVER UNFAIR IN MY FAVOR?
WATERSON



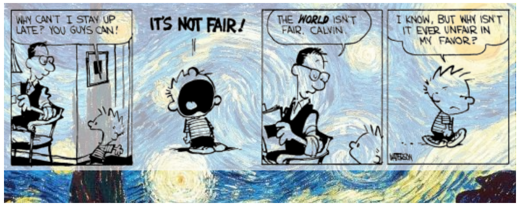

WHY CAN'T I STAY UP LATE? YOU GUYS CAN!
IT'S NOT FAIR!
THE WORLD ISN'T FAIR, CALVIN.
I KNOW, BUT WHY ISN'T IT EVER UNFAIR IN MY FAVOR?
WATERSON



Project Overview

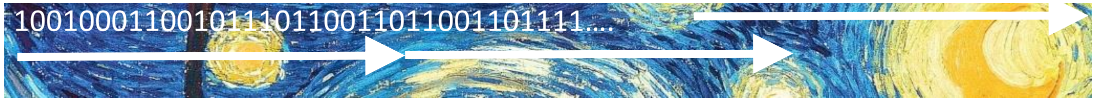
1. Conceal and reveal an image w. linear embedding

   Image → B/W image → Embed directly


WHY CAN'T I STAY UP LATE? YOU GUYS CAN!
IT'S NOT FAIR!
THE WORLD ISN'T FAIR, CALVIN.
I KNOW, BUT WHY ISN'T IT EVER UNFAIR IN MY FAVOR?
WATTERSON

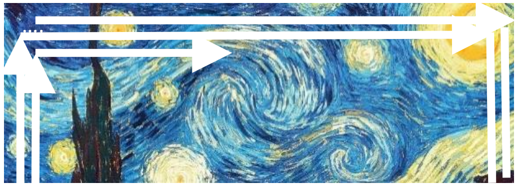
2. Conceal and reveal a text

   String → ASCII codes → Bit-sequence → Embed linearly



3. Spiral embedding of an image

   Image → Bit-Sequence → Embed in spiral



Project Overview

1. Conceal and reveal an image w. linear embedding

   Image → B/W image → Embed directly

   ImageMessage.java Steganography.java

2. Conceal and reveal a text

   String → ASCII codes → Bit-sequence → Embed linearly

   TextMessage.java Steganography.java

3. Spiral embedding of an image

   Image → Bit-Sequence → Embed in spiral

   ImageMessage.java Steganography.java

Handling Multiple Files (Classes)

- Up to now all you programs were contained in a single file.
- In this project you will be using **several files**
  - Given a static method m1() defined in a file A.java, and a static method m2() defined in a file B.java,
  - If you want to call m2 in the body of m1 you must use the following syntax; B.m2();
- E.g., in Main.java:

```
...
int[][] gray   = ImageMessage.toGray(message);
boolean[][] bw = ImageMessage.toBW(gray, 240);
int[][] hidden = Steganography.embedBWImage(cover, bw);
...
```

Provided Code (1)

```
class Helper
```

- Read and write images to two-dimensional integer array

```
public static int[][] read(String path)
public static boolean write(String path, int[][] array)
```

- Display image

```
public static void show(int[][] array, String title)
```

Image will pop-up and program will be paused until image is closed.

- Example:

```
int[][] image = Helper.read(“calvin.png");
Helper.show(image, "Original");
```

Provided Code (2)

`class Main and *Main`

- Examples of how to use the methods to hide a message in a cover image and reveal it again.

`class SignatureChecks`

- Checks that the signatures of the required methods are correct (to simplify automatic testing).
- Does not check any functionality!

`class Utils`

- Methods to checks that input (or output) data are correct, e.g., a two-dimensional array is an image.
- Helpful for debugging!

Provided Code (3)

```
class Tests
```

- Some Junit tests to simplify debugging
- These tests are **not exhaustive**, i.e., if you pass all of the tests, it does not mean that you will get full marks!
- During grading we will run automatic tests with random inputs on your submission.
- It is your responsibility to test your implementation!

Outline

- Administrative
  - Information/Starting point
  - Submission and Groups
  - Submission Server and Tokens
- Project
  - Goal
  - Overview
  - Provided Code
  - Project Details:
    - Representation of Images
    - Part 1: Concealing an image
    - Part 2: Concealing a text
    - Part 3: Spiral encoding

Representation of Images

- Digital image = raster of pixel (or picture elements)
- Resolution = number of pixels used to represent an image, e.g., 1024x768 means
  - 1024 pixels from left to right
  - 768 pixels from top to bottom
- In this project: images are represented as two-dimensional arrays (of integers or booleans)

Refresher: Arrays in Java

Example	Functionality
`image.length`	Length of an array (height of image = no of rows)
`image[4]`	Access the element at position 4 **Recall**: first element is at position 0; last element is at position length-1
`image[4].length`	Length of element at position 4 (width of row 4)
`image[4][1]`	Access to element at row 4 and column 1
`new boolean[7]`	Create a new 1-dim. boolean array with 7 entries (0-6)
`new int[4][5]`	Create a new 2-dim. integer array with 4 rows (0-3) and 5 columns (0-4)

Example	Functionality
`Arrays.copyOf(msg, msg.length)`	Copies the specified array, truncating or padding with false (if necessary) so the copy has the specified length.
`Arrays.copyOfRange(message,0,10)`	Copies the specified range of the specified array into a new array.

Color Images (RGB Values)

- Each pixel has a color defined by an RGB (Red-Green-Blue) value.
- The RGB color model is an additive color model in which red, green, and blue light are added together in various ways to reproduce a broad array of colors.
    - Each RGB value is represented by three Bytes (3x8 bits), one Byte for each color.
    - Each base colors can have an intensity between 0 (min) and 2^8-1=255 (max) in decimal, or equivalently from 00 to ff in hexadecimal.
    - In JAVA, the RGB value is stored as integer.

Refresher: Numbers in Java

- Decimal (base 10):

```
int decValue = 13;
```

- Binary (base 2: 1bit):

```
int binValue1 = 0b00000000000000000000000000001101; //32-bits
int binValue2 = 0b1101; //leading zeros are not required
```

Starting with Java 7 you can use underlines for readability. Underlines are optional.

```
int binValue1 = 0b00000000_00000000_00000000_00001101;
```

- Hexadecimal (base 16: 4bits):

```
int hexValue1 = 0x00_00_00_0d;
int hexValue2 = 0xd; //leading zeros are not required
```

- Color in JAVA: integer (4 bytes = 32 bits), e.g.,

Color	Unused/alpha	Red	Green	Blue
In binary	00000000	00100000	11000000	11111111
In hexad.	00	20	c0	ff
In decimal	0	32	192	255

Task 1: Conceal & Reveal Image

1. Convert image to black-white image (1 bit/pixel)

2. Embed black-white image into cover image

3. Retrieve black-white image from cover image

4. Convert black-white image to RGB image

Task 1: Conceal & Reveal Image

1. Convert image to black-white image (1 bit/pixel)
   - Extract red, green, blue values from RGB value
   - Convert to grey (average of red, green, blue values)
   - Convert to black-white (boolean) value given a threshold
2. Embed black-white image into cover image
   - Embed one bit (boolean) into one pixel (integer)
3. Retrieve black-white image from cover image
   - Reveal LSB (boolean) from one pixel (integer)
4. Convert black-white image to RGB image
   - Merge red, green, blue values into RGB value

Task 1: Conceal & Reveal Image

1. Convert image to black-white image (1 bit/pixel)

   Extract bits of integer

   - Extract red, green, blue values from RGB value
   - Convert to grey (average of red, green, blue values)
   - Convert to black-white (boolean) value given a threshold

2. Embed black-white image into cover image

   Manipulate bits of integer

   - Embed one bit (boolean) into one pixel (integer)

3. Retrieve black-white image from cover image

   Extract bits of integer

   - Reveal LSB (boolean) from one pixel (integer)

4. Convert black-white image to RGB image

   Manipulate bits of integer

   - Merge red, green, blue values into RGB value

Selecting Bits from Integer

start end

```
int val1 = 0b00000000_10000000_00001010_00000001;
```

- Step 1: shift right >>

```
int val2 = val1 >> 9;
```

Output: `0b00000000_10000000_0000101`

- Step 2: mask &

```
int val3 = val2 & 0b111;
```

Output: 0b~~00000000_00000000_0000~~101

0b101

Merging Bits into Integer

```
int val1 = 0b1010;
int val2 = 0b10000001;
```

Goal: `0b1010_10000001`

- Step 1: shift left <<

```
int val1_sl = val1 << 8;
```

Output in binary: `1010_00000000`

- Step 2: bitwise-or |

```
int val3 = val1_sl | val2;
```

Output in binary: `1010_10000001`

Outline

- Administrative
    - Information/Starting point
    - Submission and Groups
    - Submission Server and Tokens
- Project
    - Goal
    - Overview
    - Provided Code
    - Project Details:
        - Representation of Images
        - Part 1: Concealing an image
        - Part 2: Concealing a text
        - Part 3: Spiral encoding

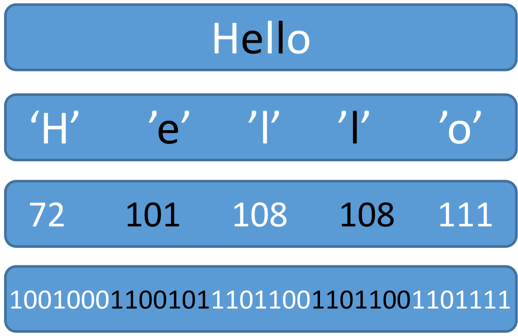
Task 2: Conceal and Reveal Text

1. Convert text into sequence of bits

2. Embed sequence of bits into cover image
3. Retrieve sequence of bits from cover
4. Convert bit-sequence to text

Task 2: Conceal and Reveal Text

1. Convert text into sequence of bits
   a) Separate String into characters
   b) Convert character to ASCII control code
   c) Convert ASCII code to 16bit-sequence



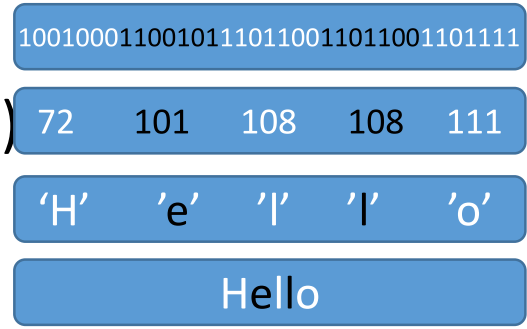
Hello

‘H’ ’e’ ’l’ ’l’ ’o’

72 101 108 108 111

10010001100101110110011011001101111

2. Embed sequence of bits into cover image
3. Retrieve sequence of bits from cover
4. Convert bit-sequence to text
   d) Separate bit-sequence into 16bit chunks
   c) Convert 16bit sequence to integer (ASCII)
   b) Convert ASCII control code to characters
   a) Merge characters to String



10010001100101110110011011001101111

72 101 108 108 111

‘H’ ’e’ ’l’ ’l’ ’o’

Hello

String and Characters in JAVA

1.a) Extract character from String: charAt

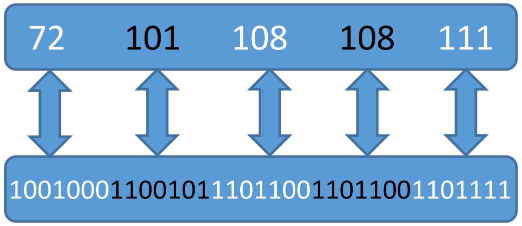
```
char c = message.charAt(i);
```

1.b) Character to integer: cast

```
int m = (int) c;
```

1.c)+ 4.c) Integer to bit-sequence & v.v.

- Review bit-manipulations from Task 1
- ICC lecture



4.b) Integer to Character: cast

```
char c = (char) m;
```

4.a) Characters to String: toString

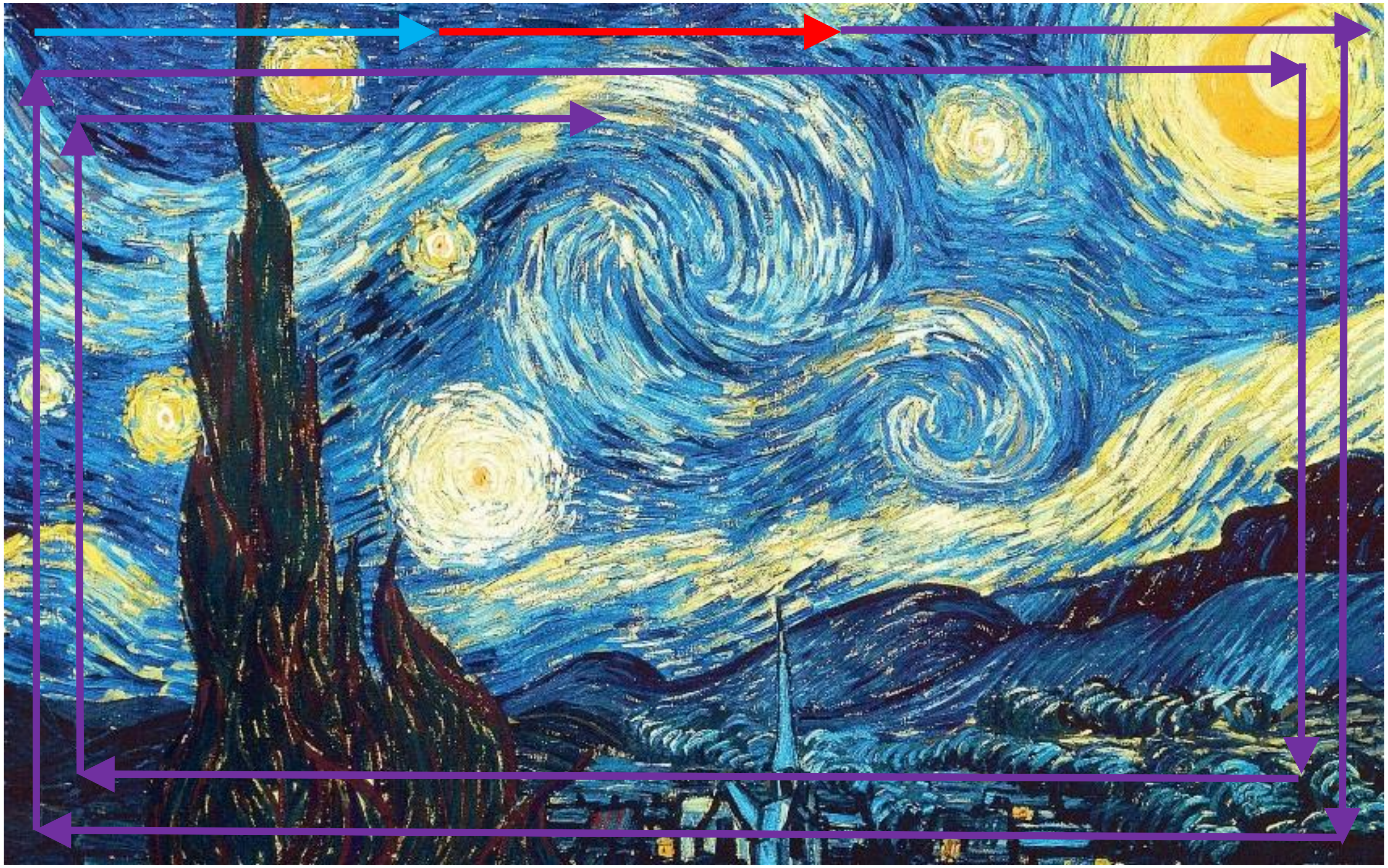
```
String message = Character.toString(c);
message = message + c;
```

Task 3: Spiral Embedding of Image

1. Convert image into bit-sequence (include size)
2. Embed bit-sequence in spiral into cover image
3. Retrieve spiral bit-sequence from cover
4. Convert bit-sequence to image

Spiral Embedding

- Bit-sequence include height, width, pixel of image

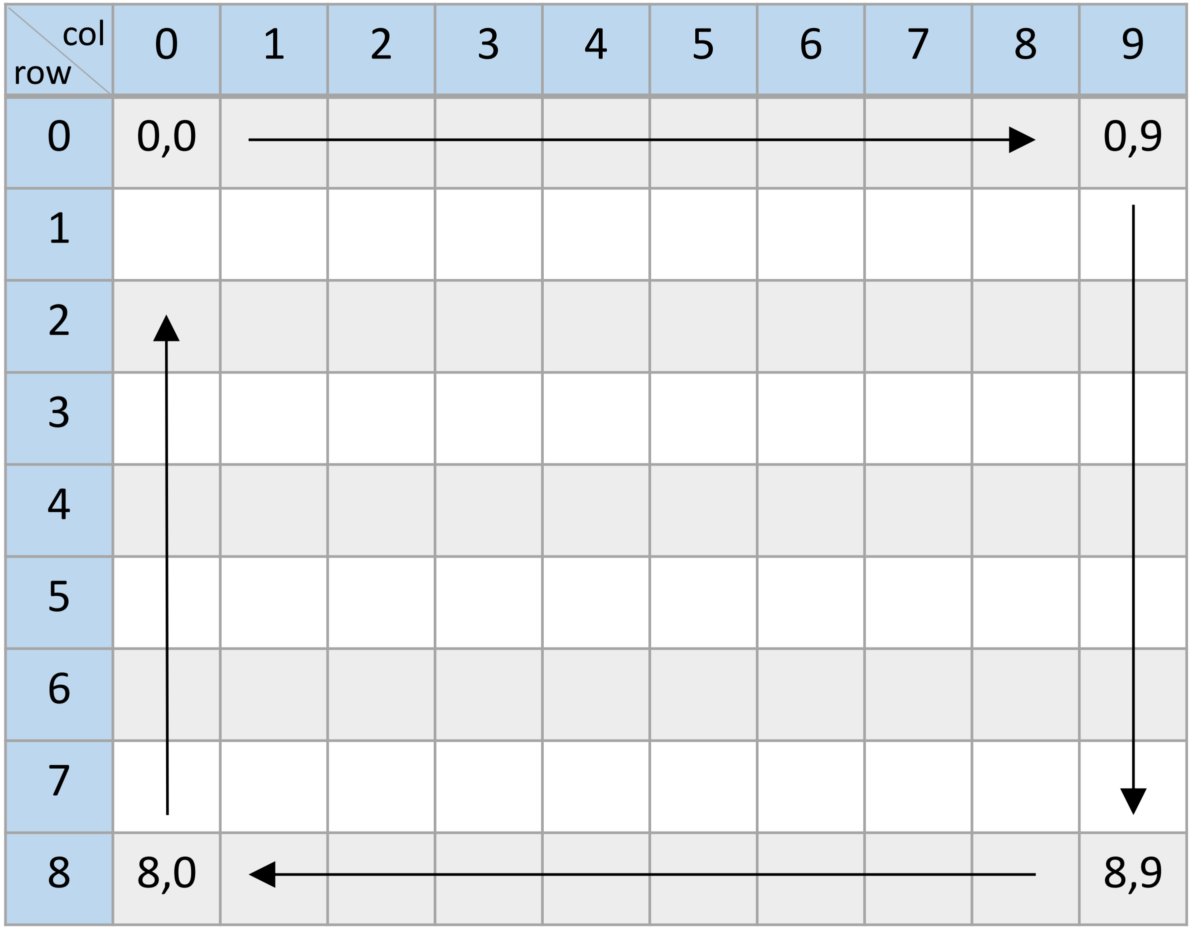


Task 3.1 & 3.4: Image ↔ Bit-Sequence

- Height and width are integers (32 bits)
    - See Task 2 for conversion of integers to bit-sequences and back
- 2-dimensional to 1-dimensional arrays and back
    - See slide 27 about handling arrays in JAVA

Task 3.2 & 3.3 Spiral Loop

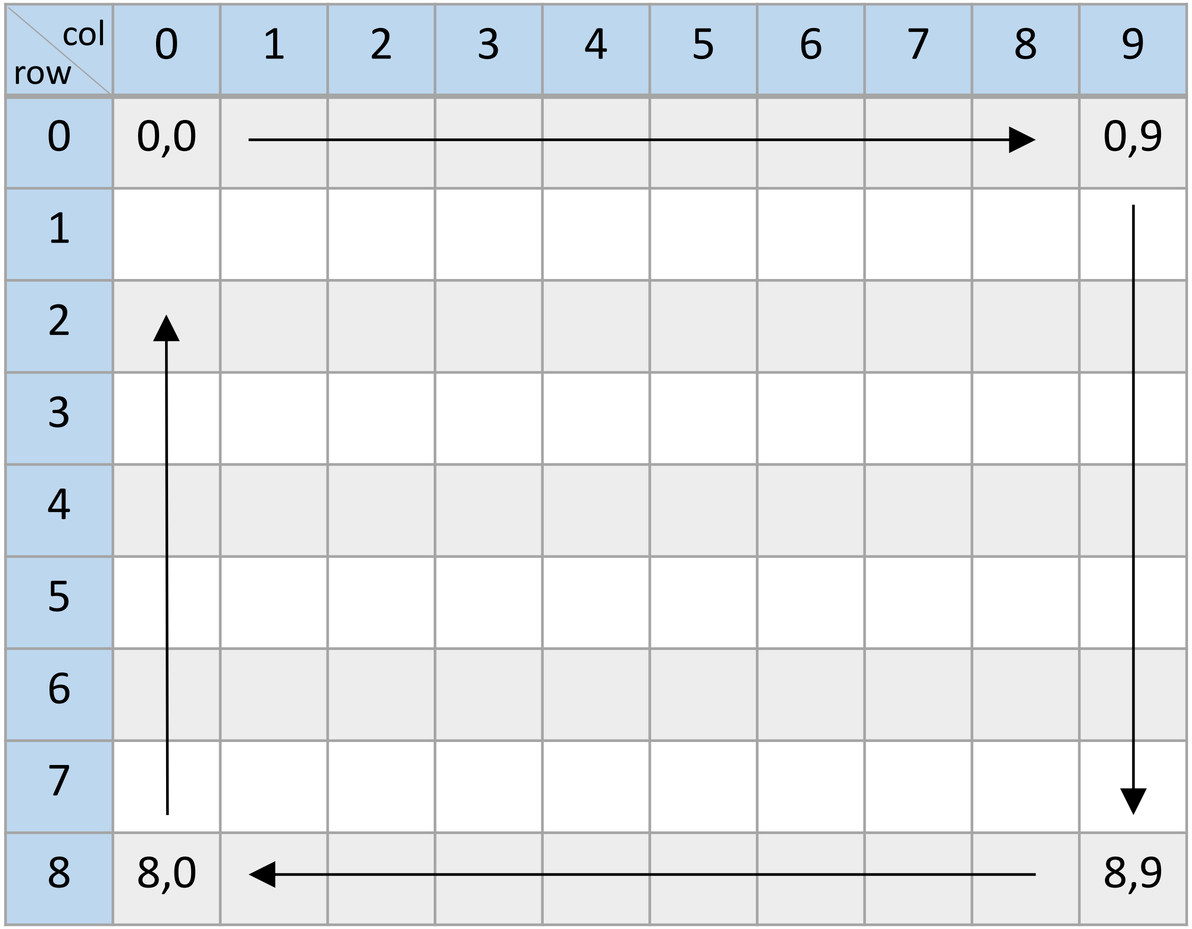
```java
int[][] image = new int[9][10];
```



col row	0	1	2	3	4	5	6	7	8	9
0	0,0									0,9
1										
2										
3										
4										
5										
6										
7										
8	8,0									8,9

Task 3.2 & 3.3 Spiral Loop

```
int[][] image = new int[9][10];
```



col row	0	1	2	3	4	5	6	7	8	9
0	0,0									0,9
1										
2										
3										
4										
5										
6										
7										
8	8,0									8,9

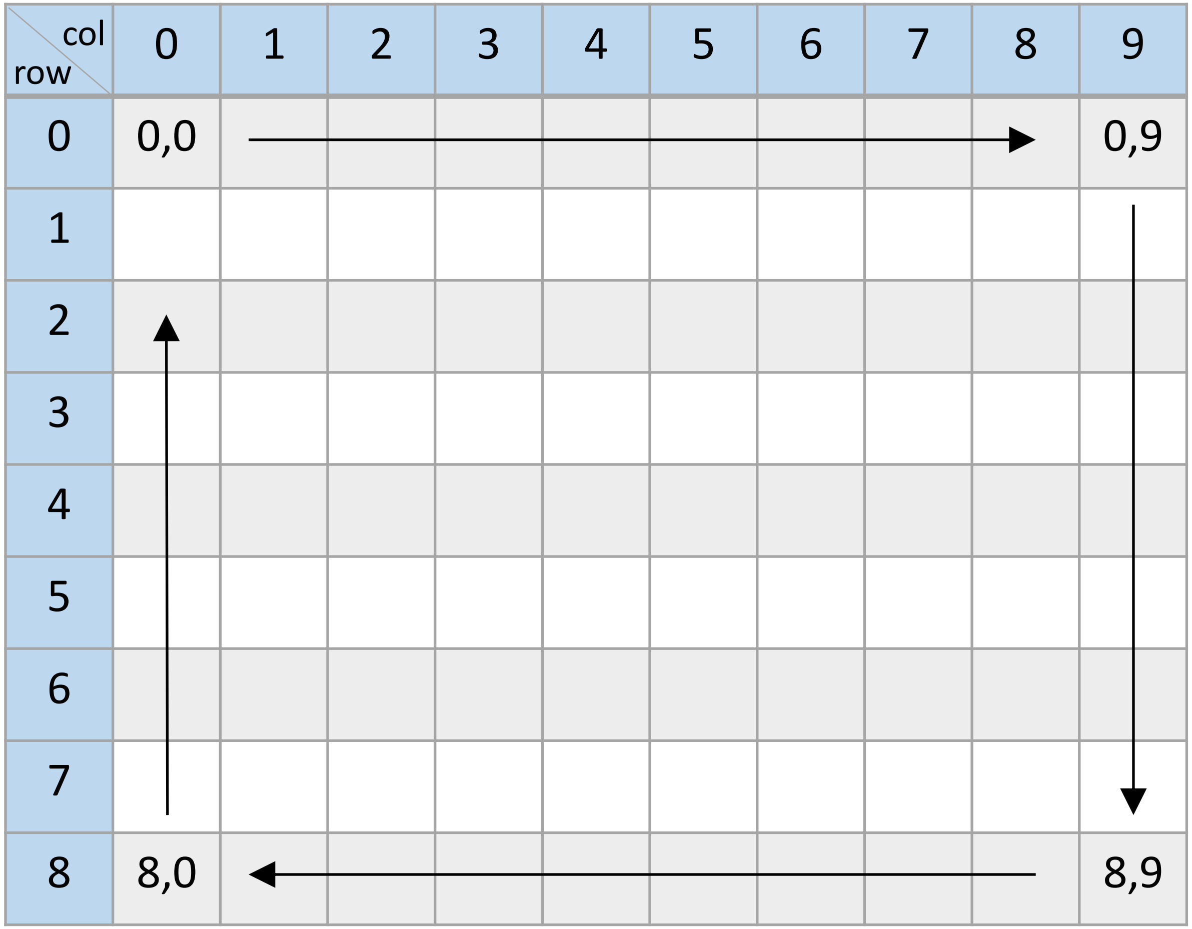
- Four Phases:
  - RIGHT
  - DOWN
  - LEFT
  - UP

```
final static int RIGHT= 0;
final static int DOWN = 1;
final static int LEFT = 2;
final static int UP   = 3;

int state = RIGHT;
```

Task 3.2 & 3.3 Spiral Loop

```
int[][] image = new int[9][10];
```

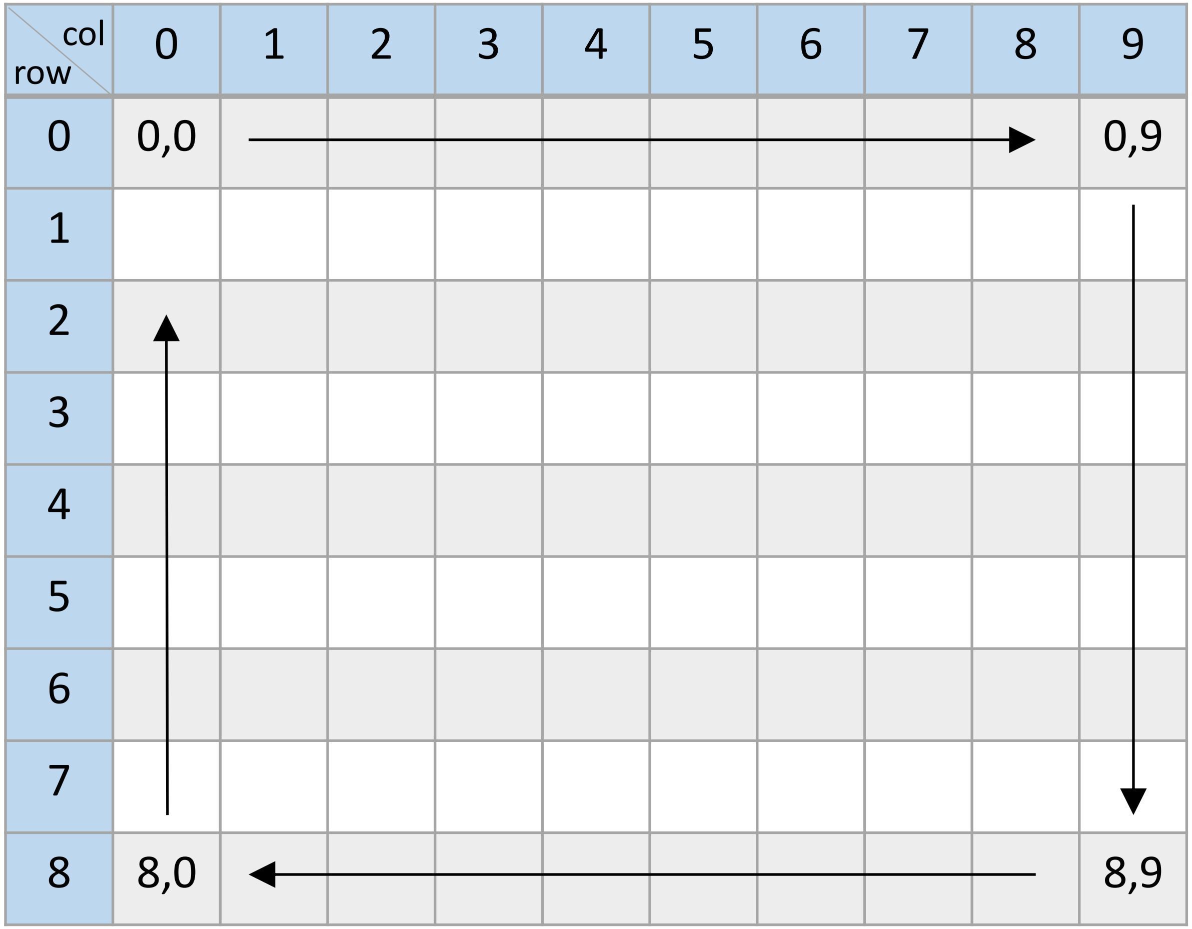


row \ col	0	1	2	3	4	5	6	7	8	9
0	0,0									0,9
1										
2										
3										
4										
5										
6										
7										
8	8,0									8,9

- Four Phases:
  - RIGHT: col++
  - DOWN: row++
  - LEFT: col--
  - UP: row—
- Transition:
  - RIGHT→DOWN
  - DOWN→LEFT
  - LEFT→UP
  - UP→RIGHT

Task 3.2 & 3.3 Spiral Loop

```
int[][] image = new int[9][10];
```

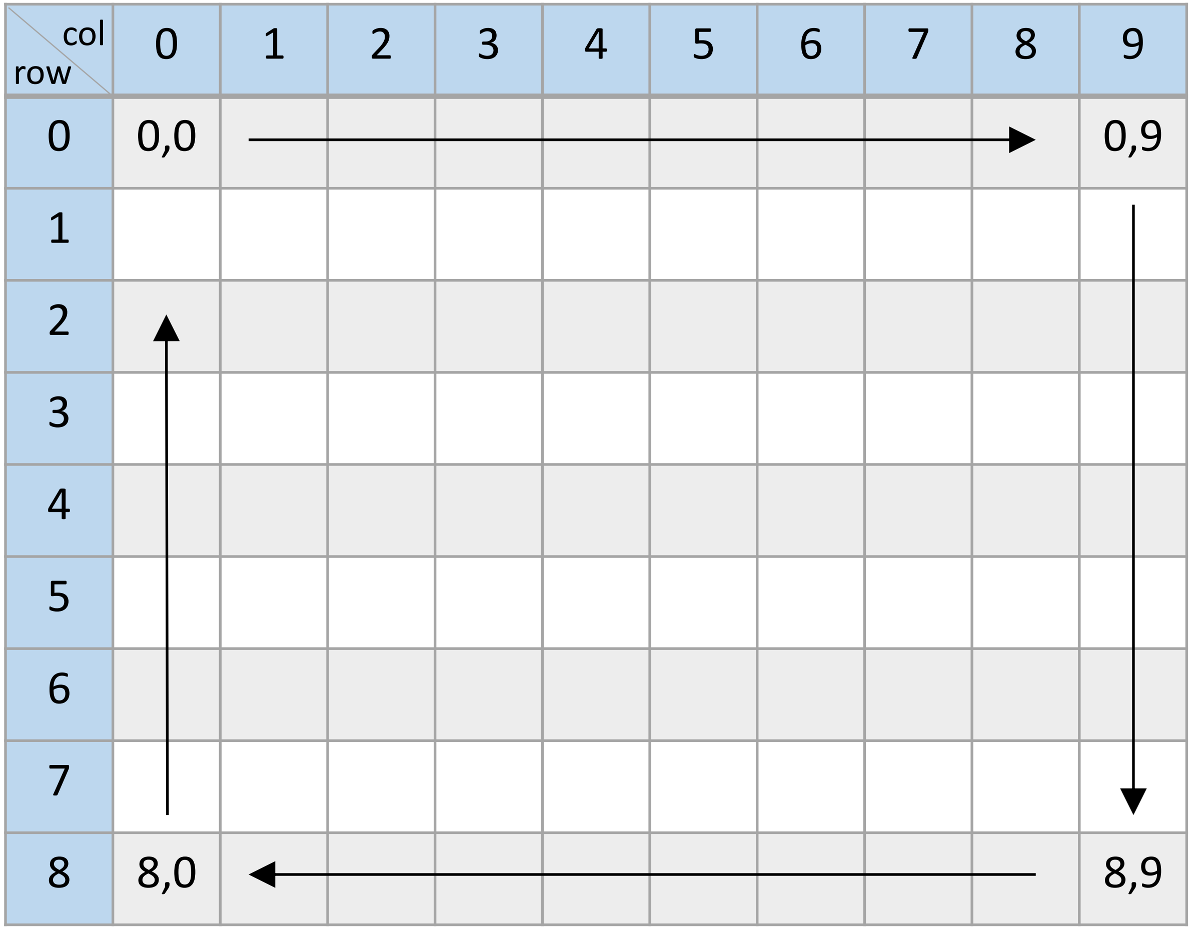


col row	0	1	2	3	4	5	6	7	8	9
0	0,0									0,9
1										
2										
3										
4										
5										
6										
7										
8	8,0									8,9

- Corner points:
  - colMin
  - colMax
  - rowMin
  - rowMax
- Initially?

Task 3.2 & 3.3 Spiral Loop

```
int[][] image = new int[9][10];
```



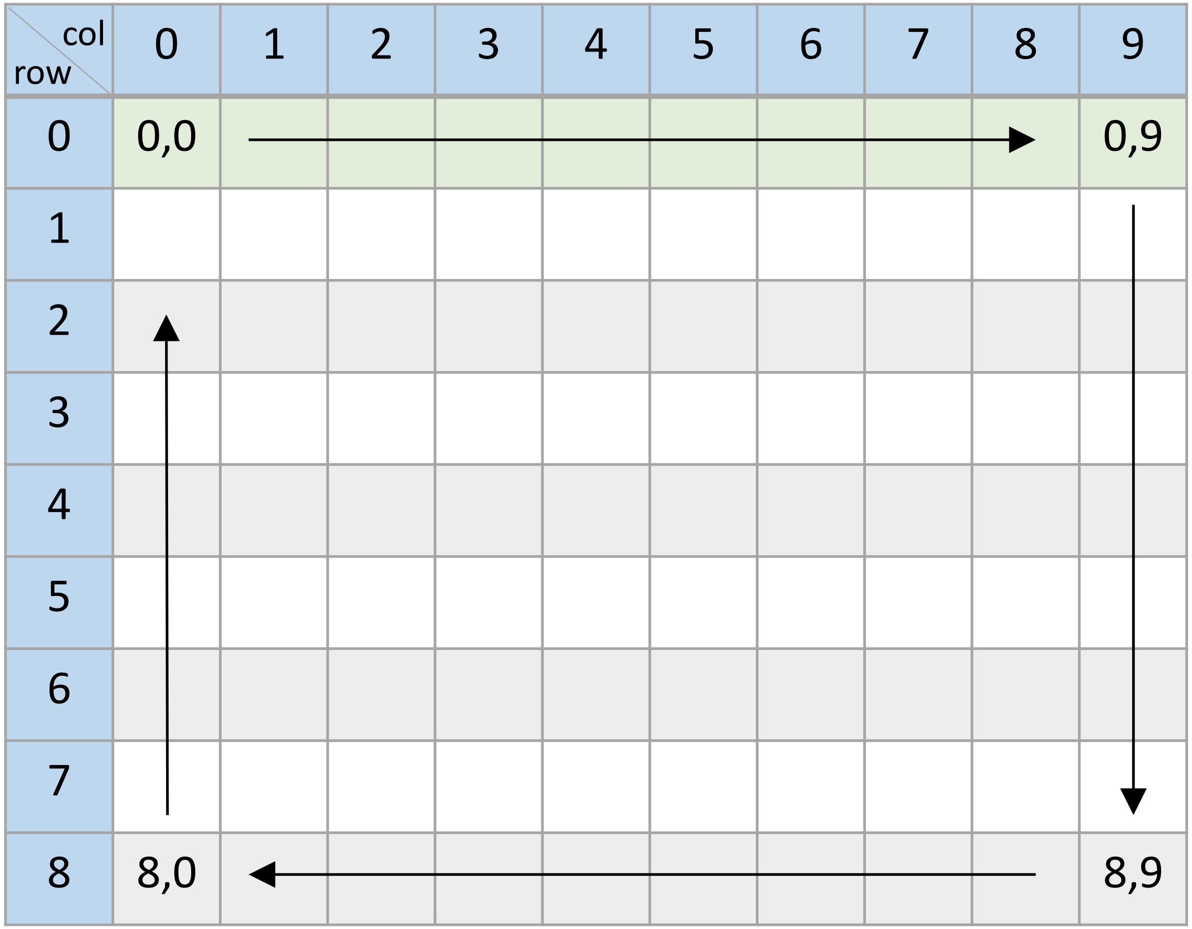
col / row	0	1	2	3	4	5	6	7	8	9
0	0,0									0,9
1										
2										
3										
4										
5										
6										
7										
8	8,0									8,9

- Corner points:
  - colMin
  - colMax
  - rowMin
  - rowMax
- Initially:

  colMin=0
  colMax=width(-1?)
  rowMin=0
  rowMax=height(-1?)

Task 3.2 & 3.3 Spiral Loop

```java
int[][] image = new int[9][10];
```



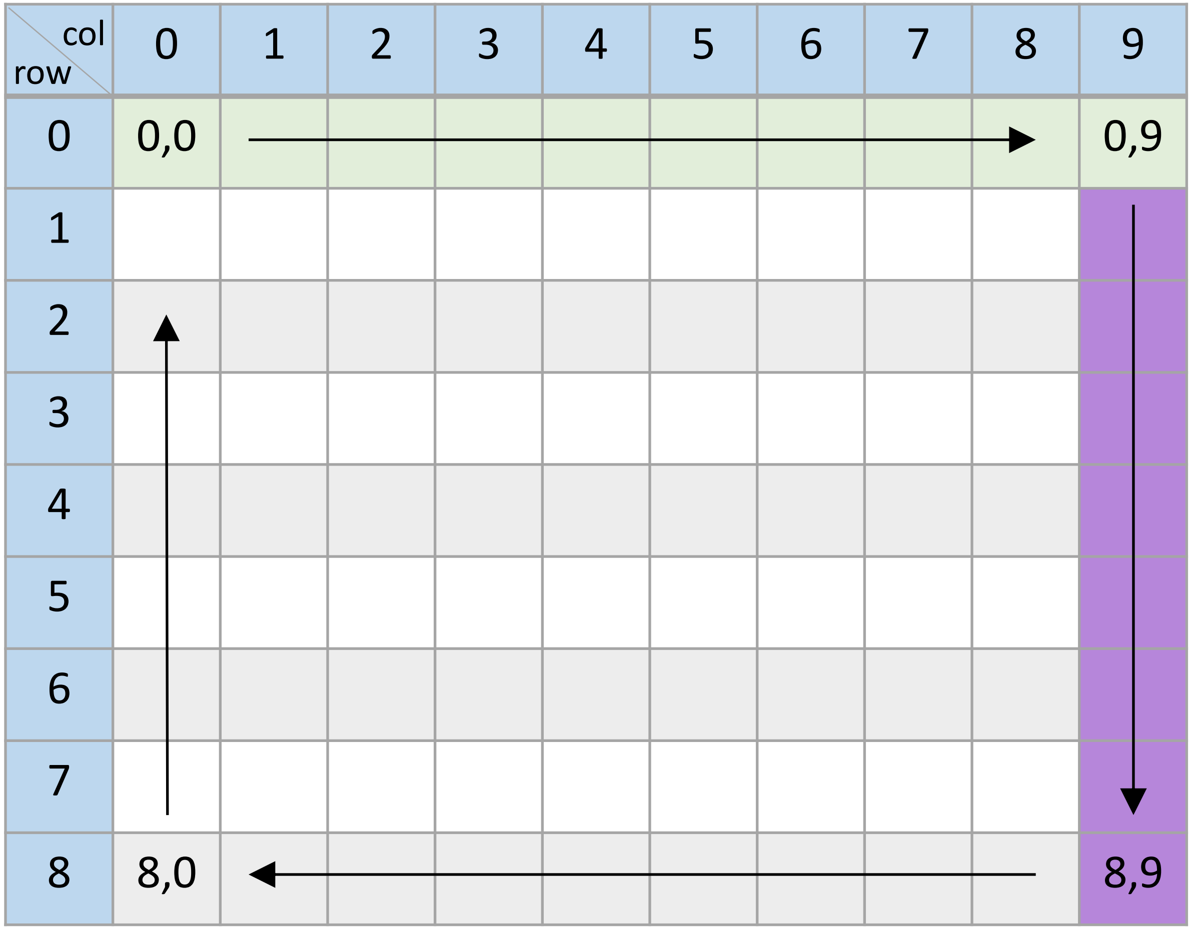
col row	0	1	2	3	4	5	6	7	8	9
0	0,0									0,9
1										
2										
3										
4										
5										
6										
7										
8	8,0									8,9

- Corner points:
    - colMin
    - colMax
    - rowMin
    - rowMax
- When to change?

  RIGHT:
  col==colMax(-1?)
  This row is done!
  rowMin++

Task 3.2 & 3.3 Spiral Loop

```
int[][] image = new int[9][10];
```



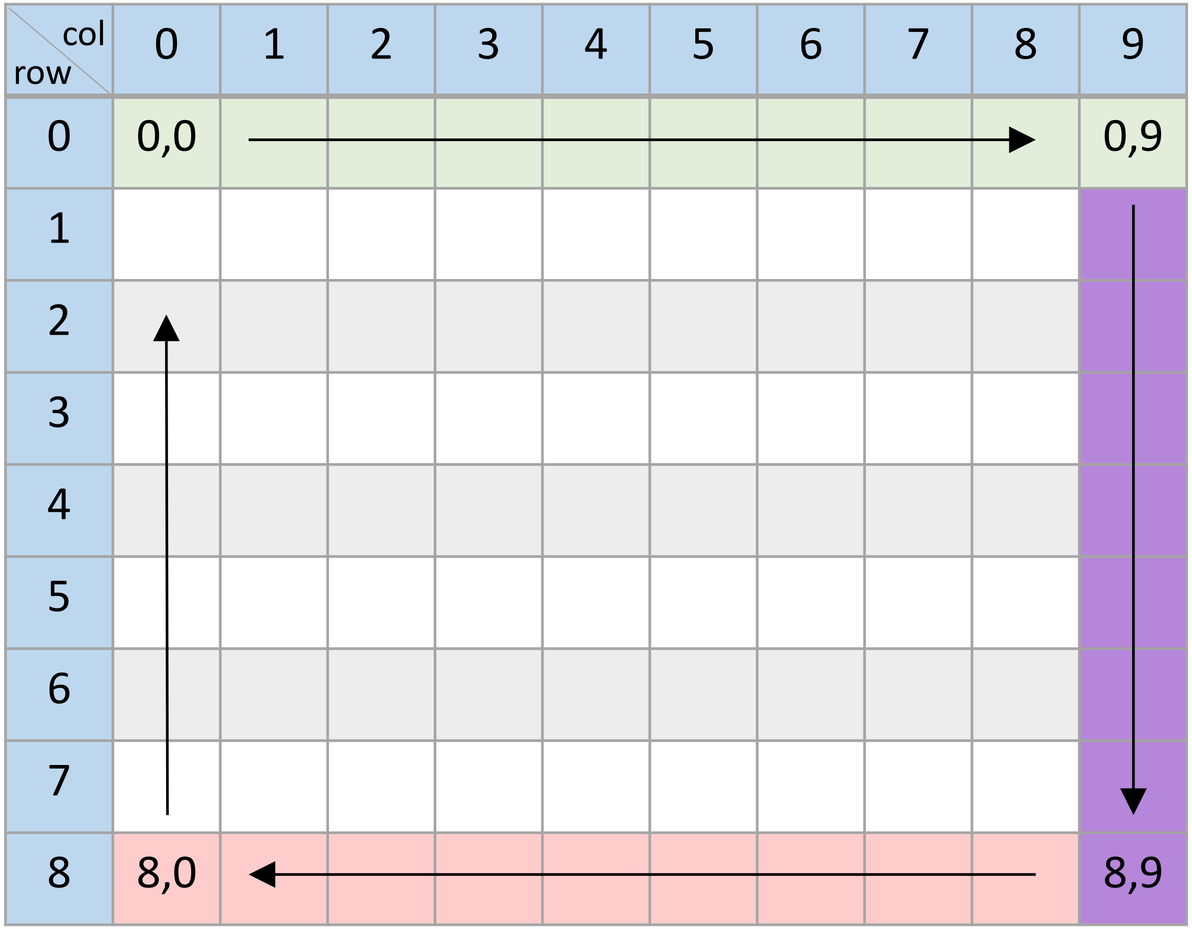
row \ col	0	1	2	3	4	5	6	7	8	9
0	0,0									0,9
1										
2										
3										
4										
5										
6										
7										
8	8,0									8,9

- Corner points:
  - colMin
  - colMax
  - rowMin
  - rowMax
- When to change?

  DOWN:
  row==rowMax
  This col is done!
  colMax--

Task 3.2 & 3.3 Spiral Loop

```
int[][] image = new int[9][10];
```



col row	0	1	2	3	4	5	6	7	8	9
0	0,0									0,9
1										
2										
3										
4										
5										
6										
7										
8	8,0									8,9

- Corner points:
  - colMin
  - colMax
  - rowMin
  - rowMax
- When to change?

LEFT:

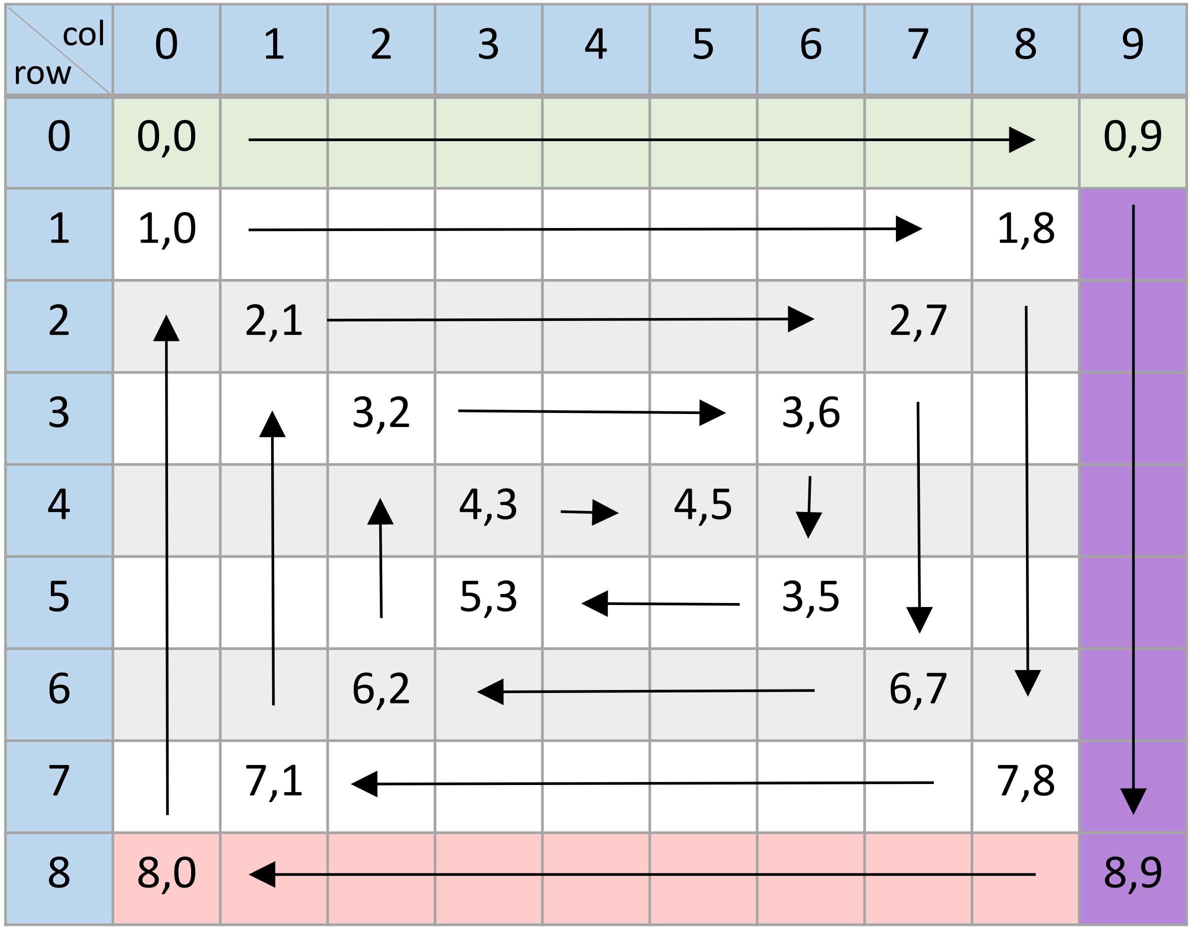
col==colMin

This row is done!

rowMax--

Task 3.2 & 3.3 Spiral Loop

```java
int[][] image = new int[9][10];
```



col row	0	1	2	3	4	5	6	7	8	9
0	0,0									0,9
1	1,0								1,8	
2		2,1						2,7		
3			3,2				3,6			
4				4,3		4,5				
5				5,3			3,5			
6			6,2					6,7		
7		7,1							7,8	
8	8,0									8,9

- Corner points:
  - colMin
  - colMax
  - rowMin
  - rowMax
- When to change?

Summary

- Administrative
  - Information/Starting point
  - Submission and Groups
  - Submission Server and Tokens
- Project
  - Goal
  - Overview
  - Provided Code
  - Project Details:
    - Representation of Images
    - Part 1: Concealing an image
    - Part 2: Concealing a text
    - Part 3: Spiral encoding