algoritmics

Module 4. Lesson 4.

The Easy Editor app. Part 3

Link to the methodological guidelines

Module 4. Lesson 4. The Easy Editor app P. 3

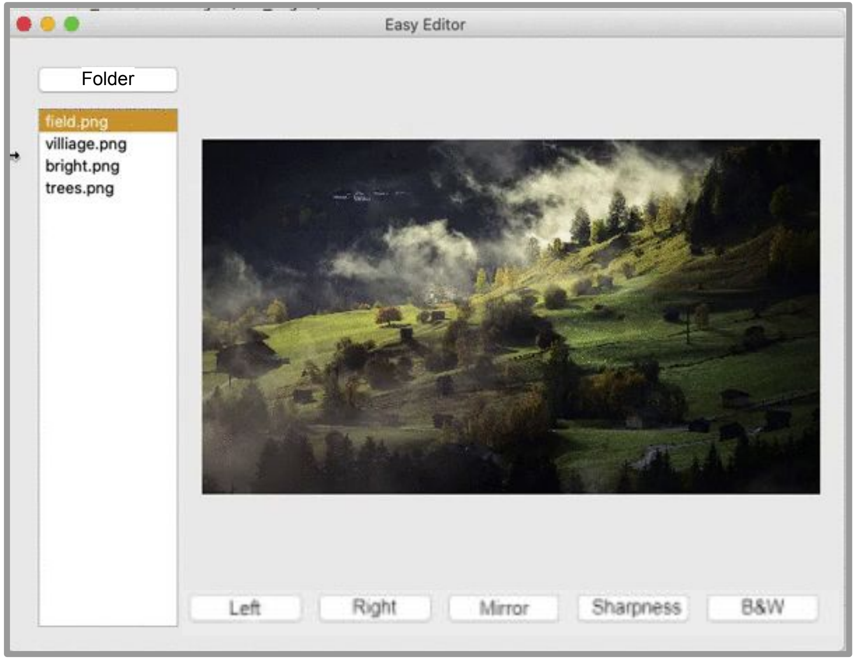
Discussion:

Project planning

Completing the order!

Today, we are going to complete our big project – **the Easy Editor photo editor**.

Let's highlight on our mind map and checklist what tasks we have for today.



Emily,
Project Manager

Discussion of tasks

Planning work on the project

Project **mind map:**

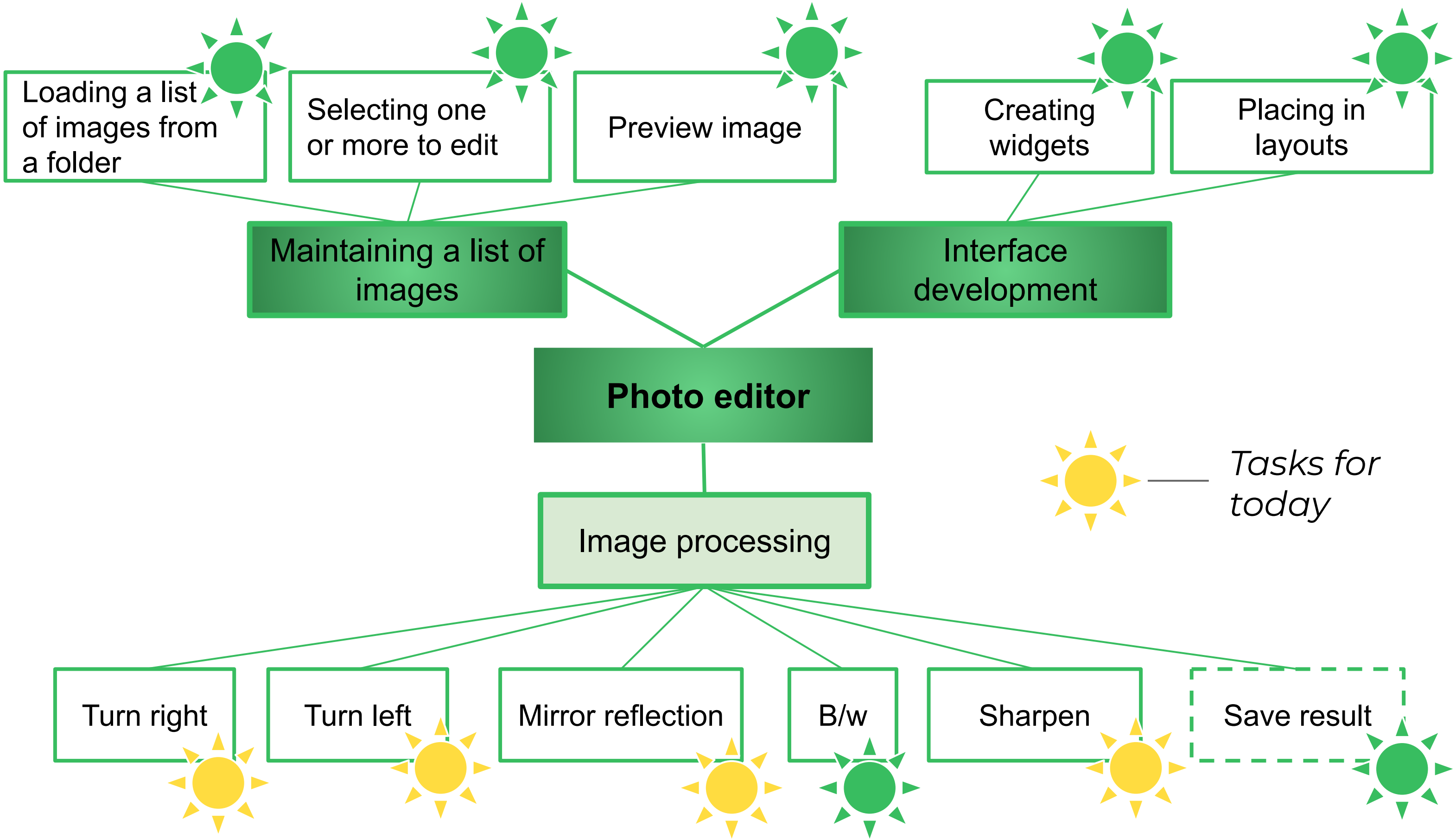


Photo editor

- Maintaining a list of images
  - Loading a list of images from a folder
  - Selecting one or more to edit
  - Preview image
- Interface development
  - Creating widgets
  - Placing in layouts
- Image processing
  - Turn right
  - Turn left
  - Mirror reflection
  - B/w
  - Sharpen
  - Save result

☀ — *Tasks for today*

Discussion of tasks

Planning work on the project

Checklist based on the **mind map**:

1. *Create an interface for* the app.
2. *Ensure loading* images from the required folder.
3. *Show a preview* of the image selected in the list.
4. *Program editing* of a photo:
   - processing tools for photos (copies of the original):
     - "Make it black and white",
     - "Rotate left (90°)",
     - "Rotate right (90°)",
     - "Sharpen"
     - "Mirror (left to right)";
   - showing a preview of the modified copy;
   - saving to the Modified subfolder.

Today

Discussion of tasks

The goal of the working day is

program image processing in the Easy Editor app.

Today you will:

- Remember and implement image processing using PIL
- Complete the Easy Editor app.
- Create test cases and assess the app’s operability (if there is enough time).

Discussion of tasks

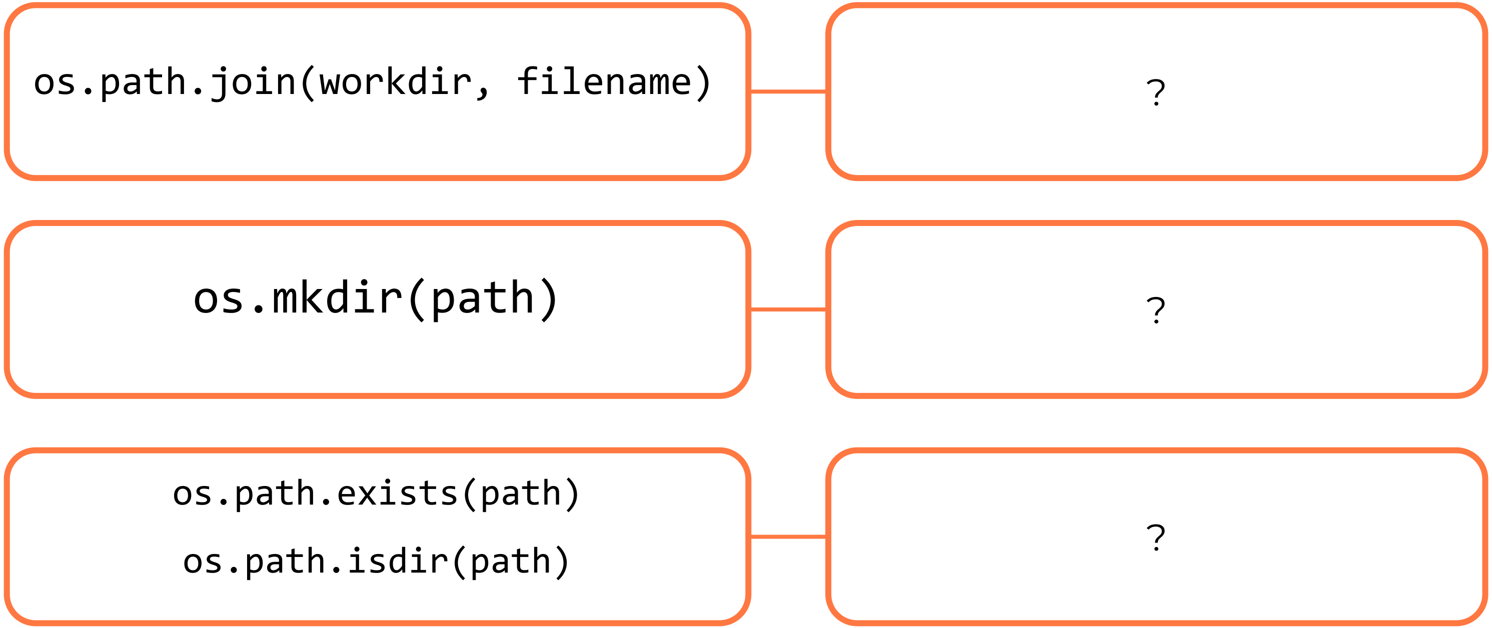
Module 4. Lesson 4. The Easy Editor app P. 3

Qualifications

Demonstrate the knowledge of PIL library and os module

Qualifications

Why do we need the os module?
Describe the purpose of the functions:



`os.path.join(workdir, filename)`

?

`os.mkdir(path)`

?

`os.path.exists(path)`

`os.path.isdir(path)`

?

Qualifications

The os module

is located in the Python standard library and contains functions for working with the operating system.

`os.path.join(workdir, filename)`	Obtaining the full path to the file by combining the path to the folder and the file name
`os.mkdir(path)`	Creating a new folder according to the specified path (the folder name is a part of the path!)

`os.path.exists(path)`
`os.path.isdir(path)`
	Check if something in this path already exists (e.g. a folder)

Qualifications

What is a path to a folder ?

And a path to a file ?

What will be the value of the cur_path variable after the programme has run:

```
cur_dir = ''
filename = 'car.png'

def chooseDir():
    global cur_dir
    cur_dir = QFileDialog.getExistingDirectory()
    return cur_dir

btn_dir.clicked.connect(chooseDir)

cur_path = os.path.join(cur_dir, filename)
```

Qualifications

A path to a folder

– is a sequence of folder (directory) names and additional characters specifying the path <u>to the folder</u>.

A path to a file

– is a sequence of folder names, characters, and the name of the file you are looking for, giving the path <u>to the file</u>.

```
cur_dir = ''
filename = 'car.png'

def chooseDir():
    global cur_dir
    cur_dir = QFileDialog.getExistingDirectory()
    return cur_dir

btn_dir.clicked.connect(chooseDir)

cur_path = os.path.join(cur_dir, filename)
```

cur_path *contains the path to the folder selected concatenated with the name of the current file.*

Qualifications

What methods of <u>processing</u> images like Image from PIL do you know?

Qualifications

Methods of image processing:

Command	*Purpose*
`from PIL import ImageFilter`	To connect the filters module
`pic_gray = original.convert('L')`	To make the image black and white
`pic_blured = original.filter(ImageFilter.BLUR)`	To blur the image
`pic_up = original.transpose(Image.ROTATE_90)`	To turn the image left 90 degrees
`pic_mir = original.transpose(Image.FLIP_LEFT_RIGHT)`	To mirror the image left to right

Qualifications

Qualifications confirmed!

Great, you are ready to brainstorm and complete the whole Easy Editor project!

Qualifications

Brainstorm:

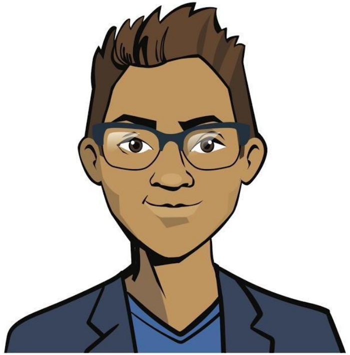
Image processing

Working tasks

Let’s complete the **ImageProcessor** class with image processing methods:

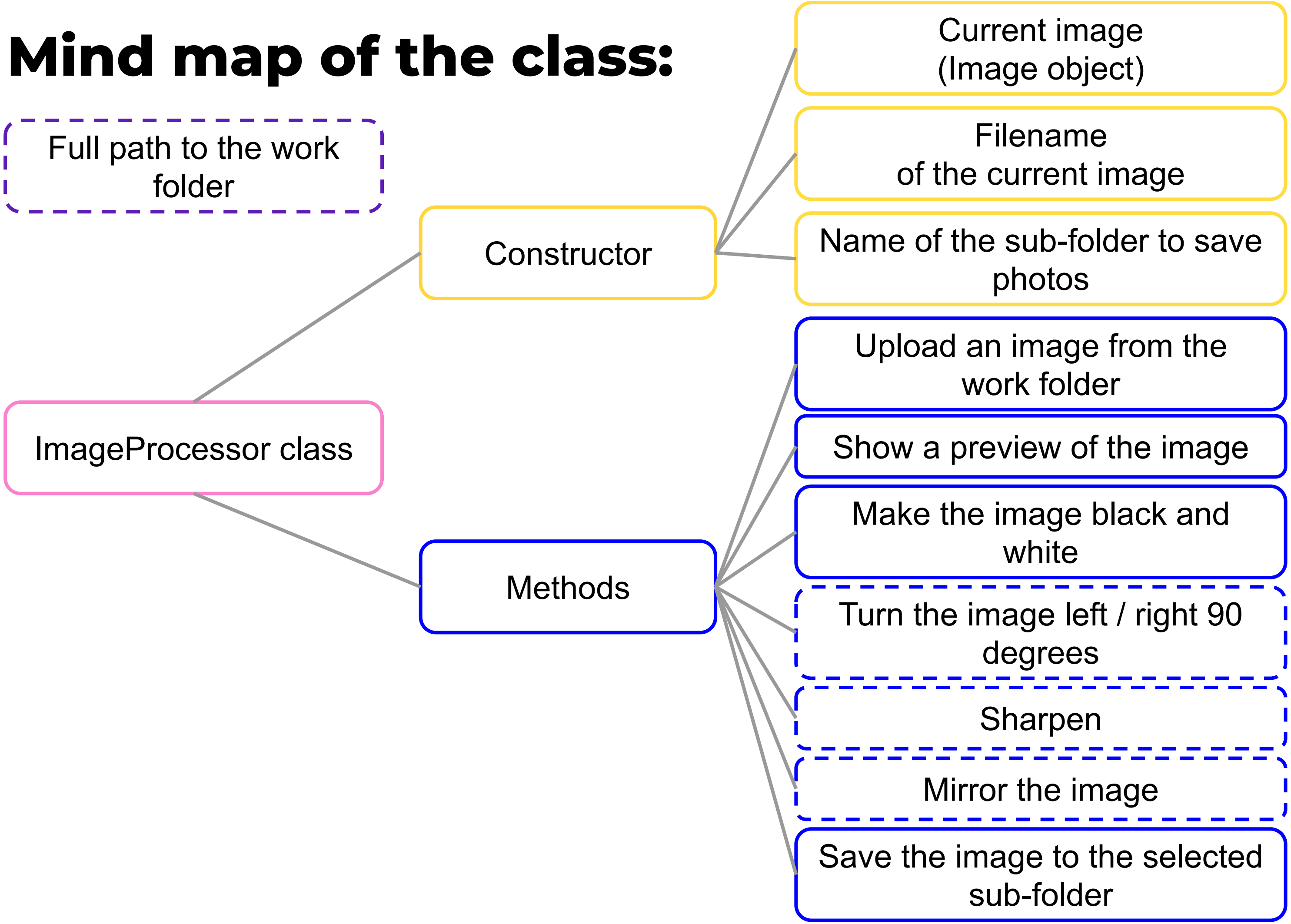
– turn the photo 90 degree lefts;
– turn the photo 90 degrees right;
– adjust the sharpness of the photo;
– mirror the image left to right.

Using these methods, we will handle clicks for the corresponding buttons.



Brainstorming

Mind map of the class:



Full path to the work folder

ImageProcessor class

Constructor

- Current image (Image object)
- Filename of the current image
- Name of the sub-folder to save photos

Methods

- Upload an image from the work folder
- Show a preview of the image
- Make the image black and white
- Turn the image left / right 90 degrees
- Sharpen
- Mirror the image
- Save the image to the selected sub-folder

Brainstorming

The ImageProcessor class: current tasks

```
class ImageProcessor():
```

constructor

- □ current **images** (defaults to None);
- □ current **filename** (defaults to None);
- □ **sub-folder name** for saving the modified images;

methods

- □ **loading the** image, **displaying the image** preview**, and saving**;
- □ **image processing:**
  - rotation;
  - sharpness;

  and other methods.

We need to program four image processing methods.

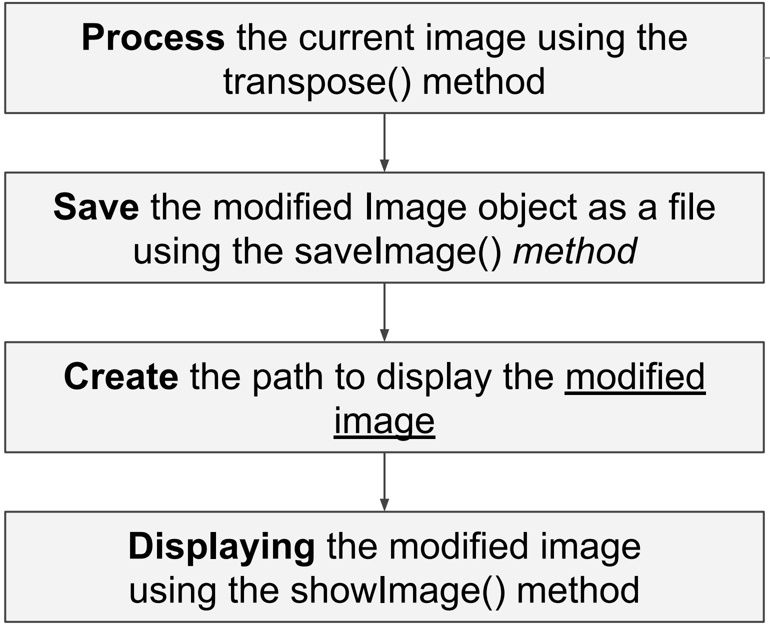
We then need to use these methods to handle the following button clicks:

```
btn_left, btn_right,
btn_flip, btn_sharp
```

Brainstorming

The do_flip() method – mirror the image

```
def do_flip(self):
```



Process the current image using the transpose() method

Save the modified Image object as a file using the saveImage() *method*

Create the path to display the modified image

Displaying the modified image using the showImage() method

We have already used the transpose() method when working on our ImageEditor training class.

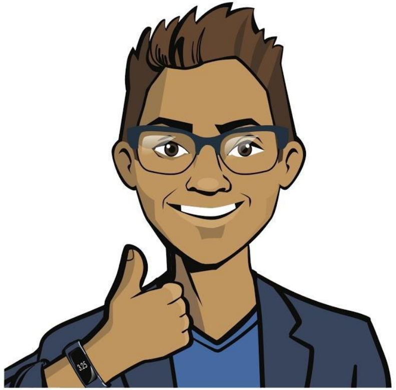
When using the **ROTATE_90** constant, it **rotates** the image.

When using the **FLIP_LEFT_RIGHT** constant, it **flips** the image left to right.

Brainstorming

The do_flip() method – mirror the image

```
def do_flip(self):
    self.image = self.image.transpose(Image.FLIP_LEFT_RIGHT)
    self.saveImage()
    image_path = os.path.join(
        workdir, self.save_dir, self.filename
    )
    self.showImage(image_path)
```

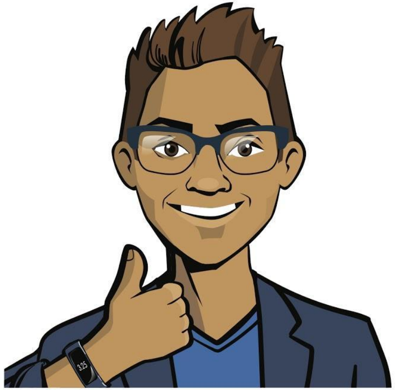


Brainstorming

The do_flip() method – mirror the image

```
def do_flip(self):
    self.image = self.image.transpose(Image.FLIP_LEFT_RIGHT)
    self.saveImage()
    image_path = os.path.join(
        workdir, self.save_dir, self.filename
    )
    self.showImage(image_path)
```

The other methods of image processing are implemented in a similar way!



Brainstorming

Implementing the solution in the project:

The described interface elements

Reading and displaying file names

```
class ImageProcessor():
```

Class description

The do_flip() method

The other processing methods

Adding new image processing methods.

```
workimage = ImageProcessor()
```

```
def showChosenImage():
```

Function body

```
lw_files.currentRowChanged.connect(showChosenImage)
btn_bw.clicked.connect(workimage.do_flip)
```

Handling the rest of the button clicks

Handling clicks on the “Mirror” using do_flip().

Brainstorming

Your task is:

To program image processing in the Easy Editor app.

Use the technical documentation from previous workdays, if needed.

Cole,
Senior Developer

Brainstorming

Module 4. Lesson 4. The Easy Editor app P. 3

VS Code:

The Easy Editor app

Complete task 5 in VS Code

The Easy Editor app

Visual Studio Code

Work on the platform

Module 4. Lesson 4. The Easy Editor app P. 3

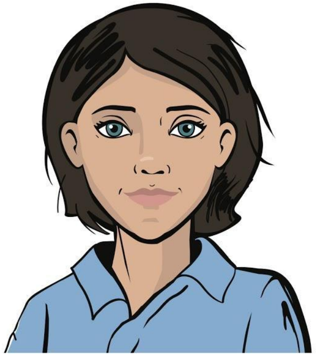
Brainstorm:

Testing an IT product

IT product life cycle

Creating an app is just one stage of the IT product life cycle.

Today, we are going to learn about and implement another important stage – **testing**.

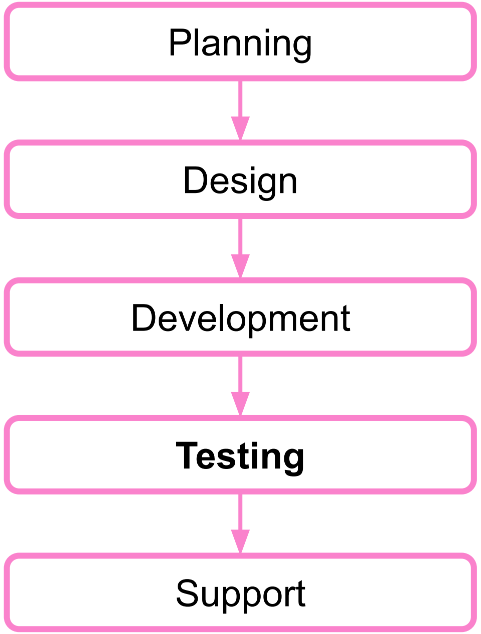


Brainstorming

IT product life cycle

A product (for example, a program) is an article of trade which can solve a significant problem or task, and, therefore, it has a value for the market (for a client).

Stages of product creation and implementation:

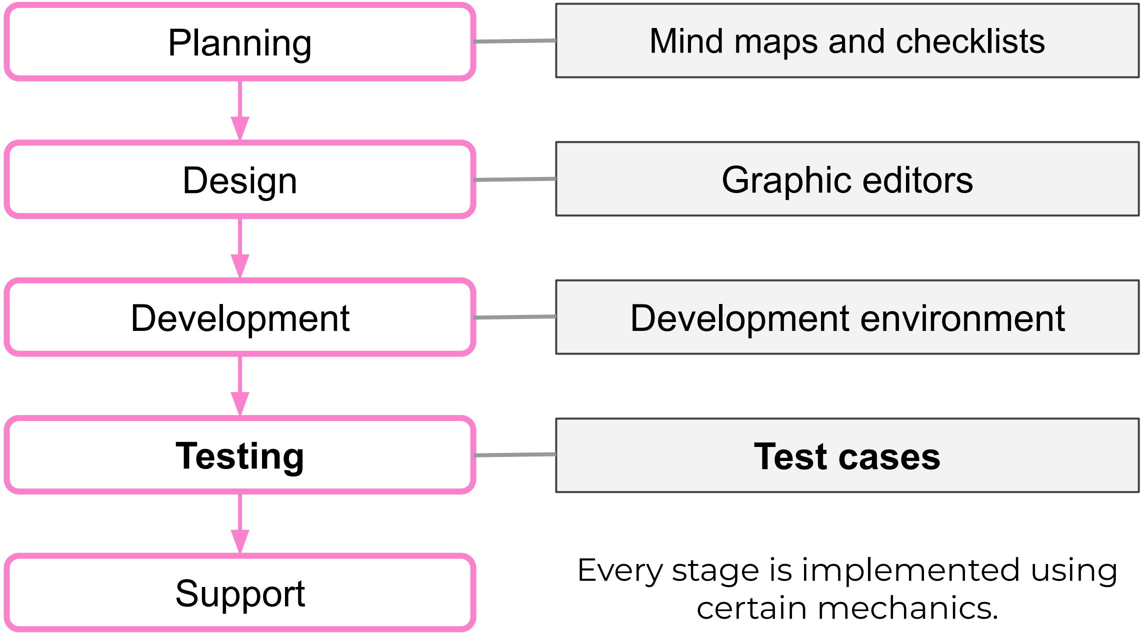


Brainstorming

IT product life cycle

A product (for example, a program) is a good which solves a significant problem, or task, and is thus valuable for the market (client).

Stages of product creation and implementation:



Planning — Mind maps and checklists

Design — Graphic editors

Development — Development environment

Testing — **Test cases**

Support

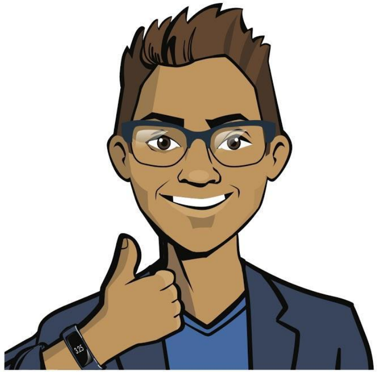
Every stage is implemented using certain mechanics.

Brainstorming

Why do we need to test an IT product?

- When developing a large project, it is difficult to notice small flaws.
- It is hard to predict all the possible actions of our users and address those in our code.

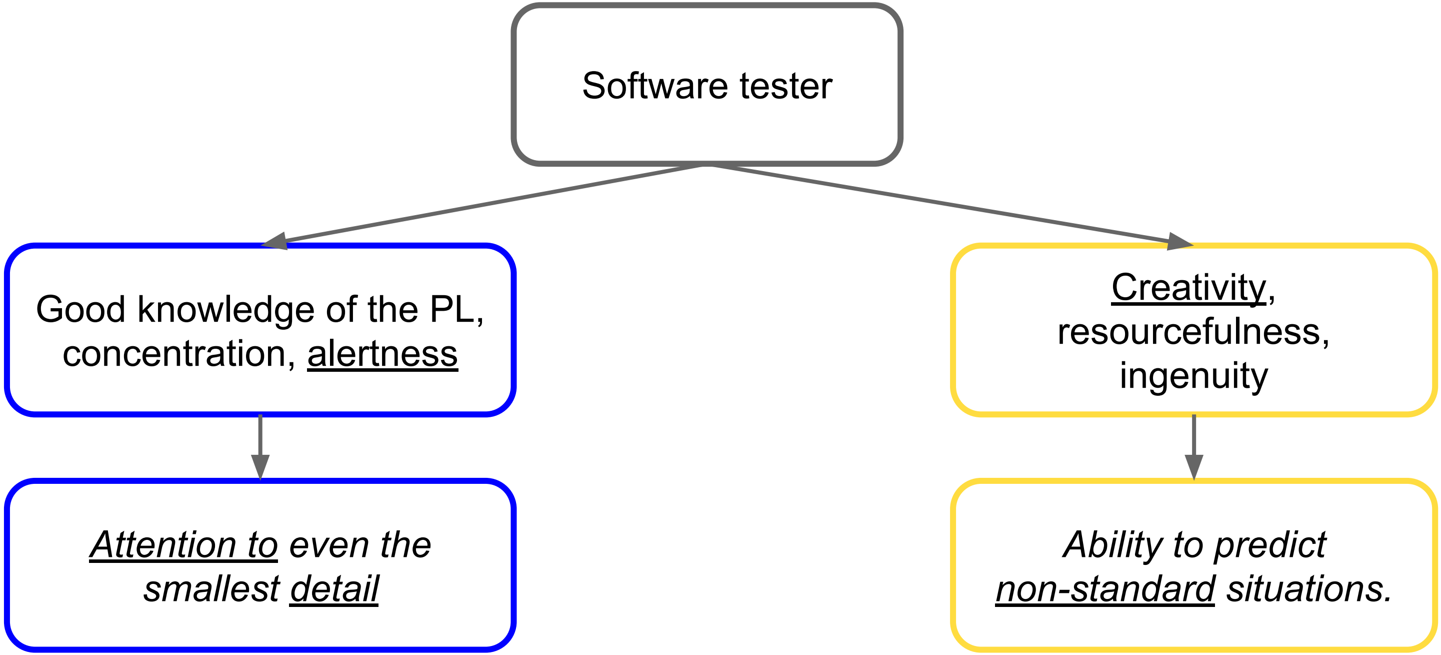
With a well-tested product, you can rest assured that your program will not suddenly break just after release!



Brainstorming

A tester

is a specialist involved in testing software to identify and correct errors.



Software tester

Good knowledge of the PL, concentration, alertness

Attention to even the smallest detail

Creativity, resourcefulness, ingenuity

Ability to predict non-standard situations.

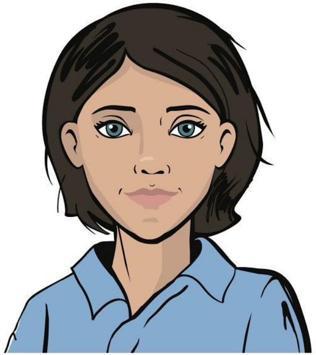
Brainstorming

Testing an IT product

One app can have several different functions. Testers check each of these functions against one or several **test cases**.

A **test case** is a document that describes the checking procedure:

- ❏ the function being tested;
- ❏ steps to reach the goal;
- ❏ expected result.



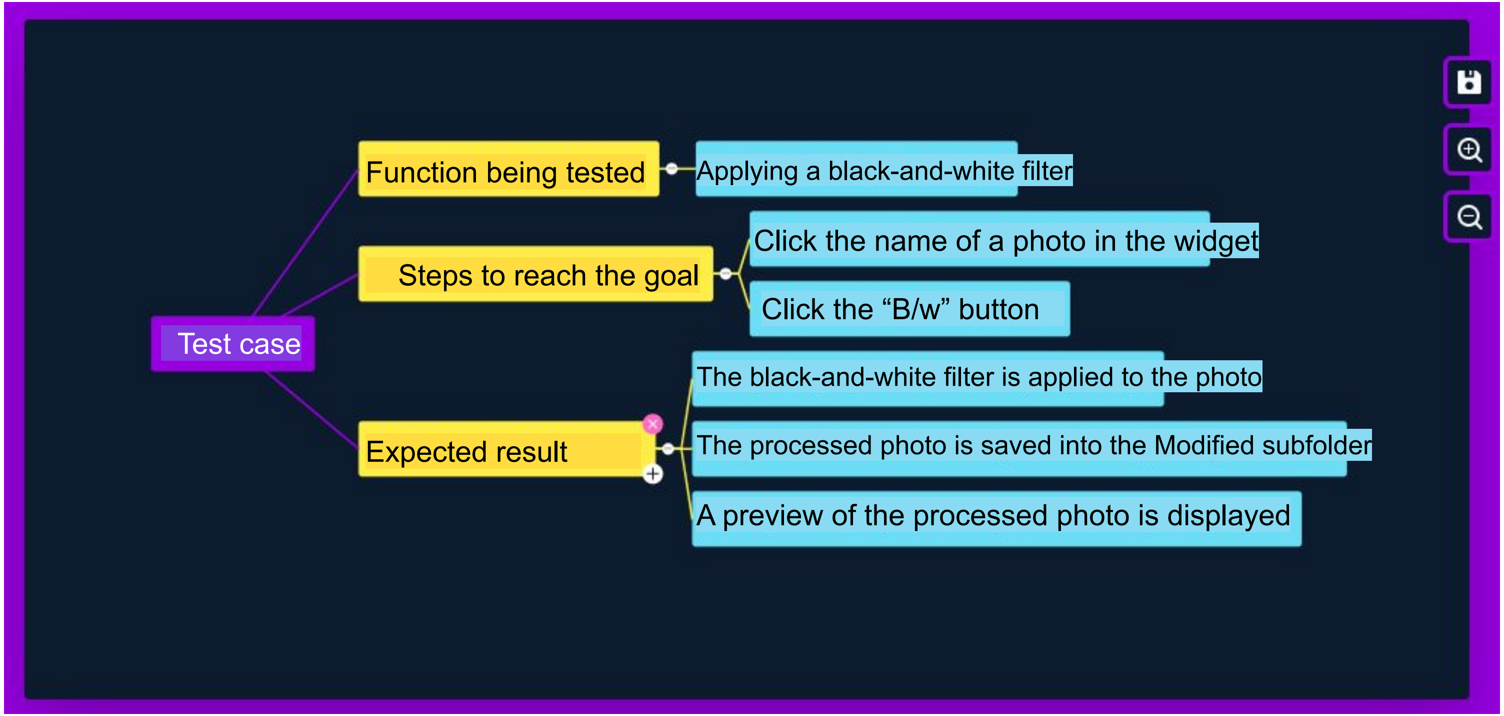
Brainstorming

“Applying a black-and-white filter” test case

Step	*Value*
The function being tested: **what goal are we pursuing?**	Applying a black-and-white filter to a photo selected from the list widget.
Steps to achieve the goal: **what does the user do?**	1. The user clicks the name of a photo in the list widget. 2. The user clicks the “B/w” button.
Expected result: **what does the program do?**	1. The black-and-white filter is applied to the photo. 2. The processed photo is saved into the Modified subfolder in the working folder. 3. A preview of the processed photo is shown in the app window.

“Applying a black-and-white filter” test case

You can use any tools to describe test cases.
For example, mind maps!



Test case

- Function being tested
  - Applying a black-and-white filter
- Steps to reach the goal
  - Click the name of a photo in the widget
  - Click the “B/w” button
- Expected result
  - The black-and-white filter is applied to the photo
  - The processed photo is saved into the Modified subfolder
  - A preview of the processed photo is displayed

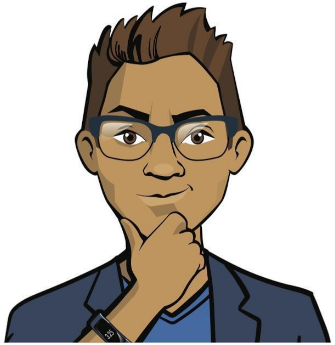
Brainstorming

Task:

Create and describe **three test cases for the Easy Editor app**.

Test your app using those cases.

Have you identified any bugs? Can you proceed to send the project to the client?



Brainstorming

VS Code:

The Easy Editor app

Complete the task on the platform

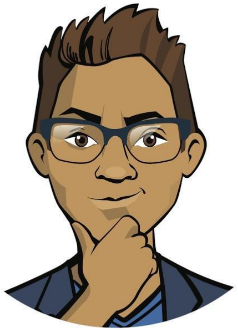
“Testing an IT product”

mars.alg.academy

Wrapping up the workday

To wrap up, pass a technical interview:

1. What stages of work on a project do you know? Which of them have you completed working on Easy Editor?
2. Who are software testers? What do they do?



Cole,
Senior Developer

Emily,
Project Manager

Wrapping up the workday

Great job!

Dear colleagues!

We congratulate you on completing the Easy Editor app!

Probably, **we will soon consider promoting you to the position of lead developer.** However, this will require you to master not only software development but other product life cycle stages as well.

Wrapping up the workday