Planning a software project

Ebba Ahlgren Cederlöf

02.12.2021

Delimitations

What is this presentation?

- An example of how to break down a big coding project into smaller steps.
- A reminder that you can save time and work if you plan ahead.

What is it not?

- A guide on how to plan every type of project
- A guide on time or risk managament

Why plan your project?

- Might not be necessary for small projects, but for larger projects it can
  - Help save you time and work
  - Help divide the work between multiple people
  - Help you identify and resolve otherwise hidden issues
  - Break down the project into approachable steps

1. Choose language and tools

- Sometimes this is already decided for you
- Scripted vs. Compiled languages (see Jana's talk)

Adapted from https://www.educative.io/blog/how-to-plan-a-coding-project

Example: Tamagotchi

1. Choose a language and tools

- Python because it is simple

2. List all features and entities

- List all features
  - Divide them into essential and nonessential to help prioritize
- List all entities that will interact with your program (e.g. users, servers, etc.)

Example: Tamagotchi

2. List all features and entities

Essential

- Create new tamagotchi
- Random mood generator
- User (re)actions:
  - Feed
  - Play
  - Scold
  - Clean
- Update stats depending on reaction.
- Death if stats drop to zero.

Non essential

- Animation
- Treat sick
- Add sound effects
- Add mini-game when ”play”

Entities

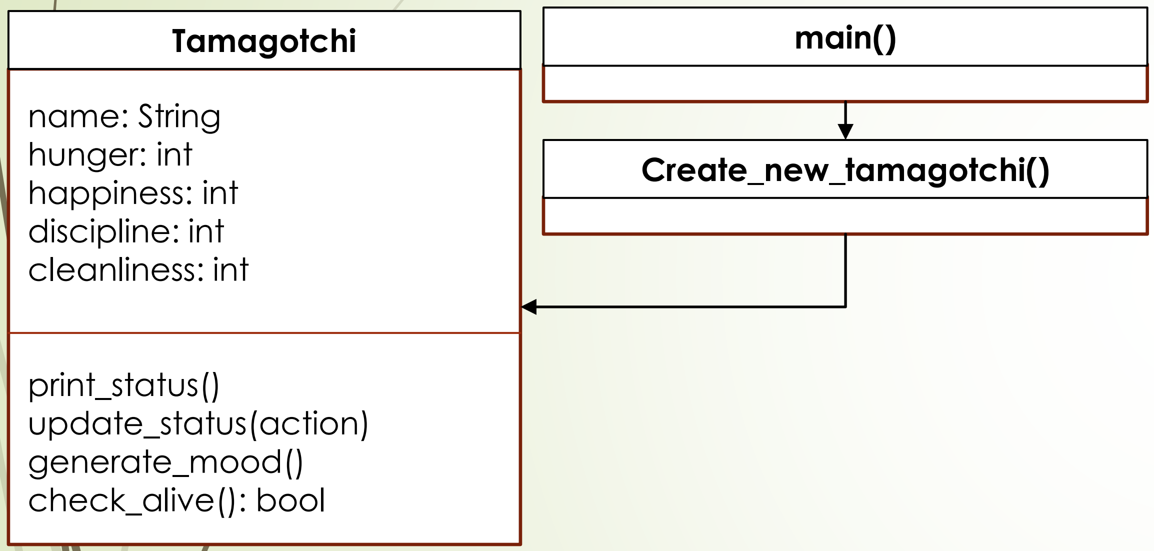
- 1 user

3. Map project architecture

- Put each feature that you wrote down into a box.
- Go through each feature and connect it to other features that it needs to interact with.
- Reveals the general skeleton of your code, might need a few iterations
- Can make it more ”flowcharty” to focus more on the flow, or more like a UML diagram to focus more on the architecture.

Example: Tamagotchi

3. Map project architecture



Tamagotchi

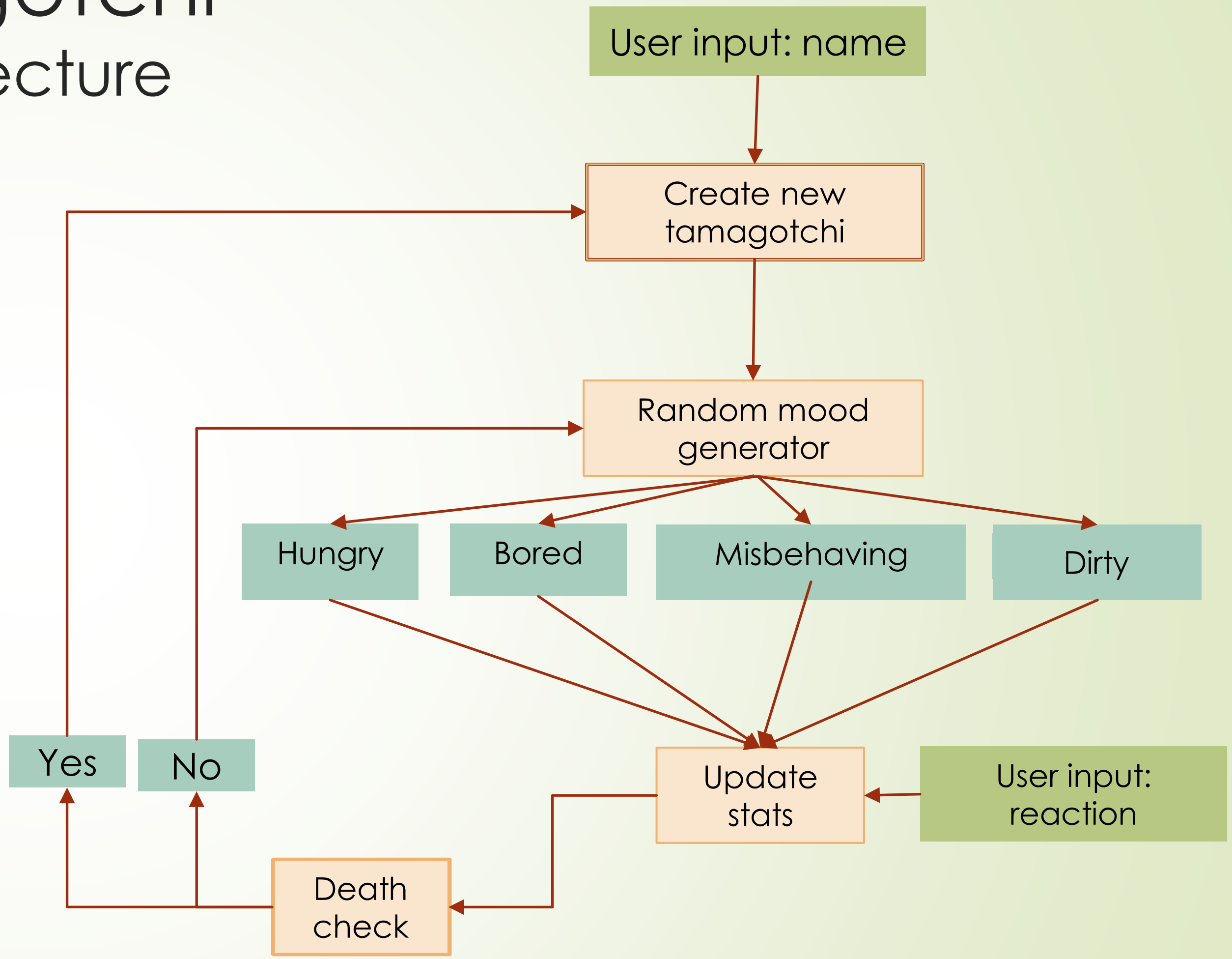
name: String
hunger: int
happiness: int
discipline: int
cleanliness: int

print_status()
update_status(action)
generate_mood()
check_alive(): bool

main()

Create_new_tamagotchi()

More UML-like



User input: name

Create new tamagotchi

Random mood generator

Hungry

Bored

Misbehaving

Dirty

Update stats

User input: reaction

Death check

Yes

No

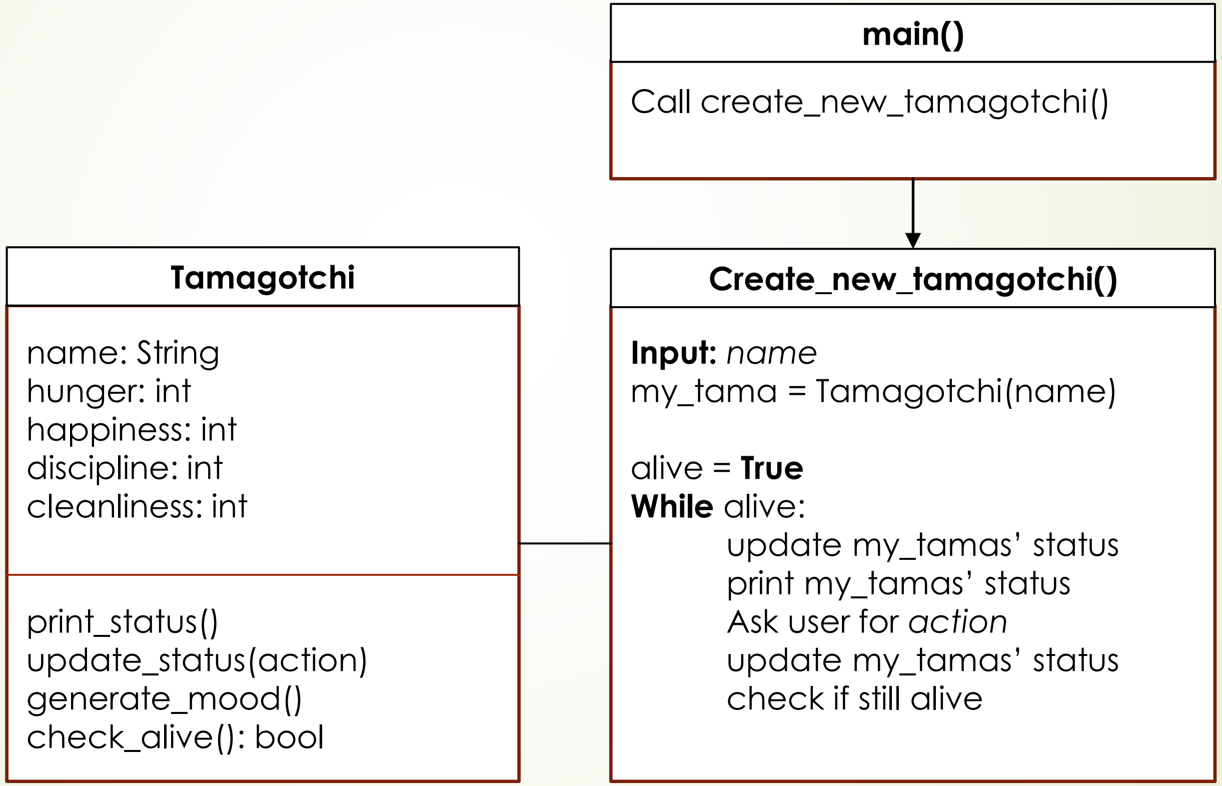
More flow-charty

4. Add pseudocode to your diagram

- When the general architecture of your program is done – add pseudo code to explain functions and algorithms. (See Jenny's talk)

Example: Tamagotchi

4. Add pseudocode to the diagram



main()

Call create_new_tamagotchi()

Tamagotchi

name: String
hunger: int
happiness: int
discipline: int
cleanliness: int

print_status()
update_status(action)
generate_mood()
check_alive(): bool

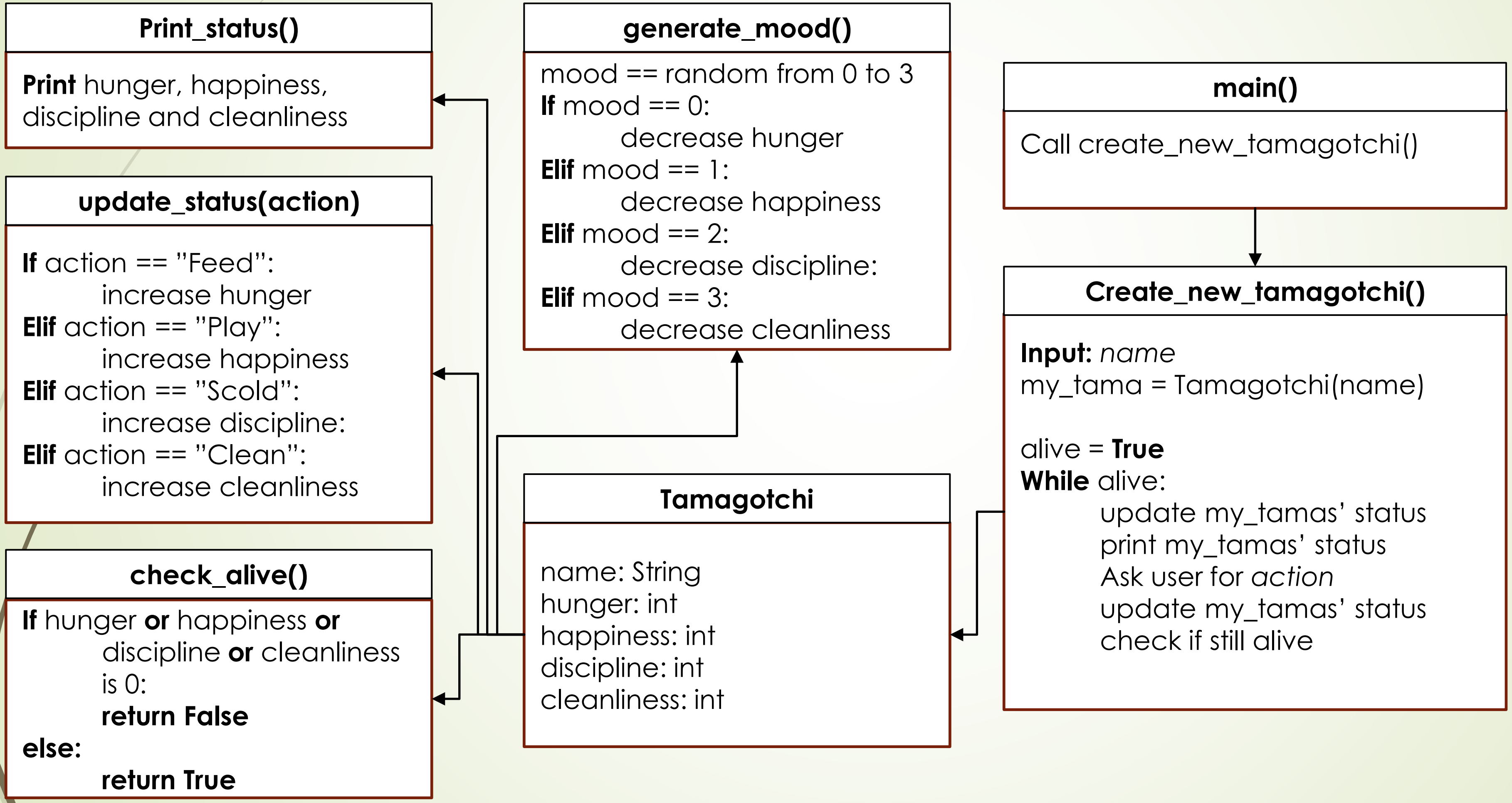
Create_new_tamagotchi()

```
Input: name
my_tama = Tamagotchi(name)

alive = True
While alive:
        update my_tamas’ status
        print my_tamas’ status
        Ask user for action
        update my_tamas’ status
        check if still alive
```

Example: Tamagotchi

4. Add pseudocode to the diagram



Print_status()

```
Print hunger, happiness,
discipline and cleanliness
```

update_status(action)

```
If action == "Feed":
        increase hunger
Elif action == "Play":
        increase happiness
Elif action == "Scold":
        increase discipline:
Elif action == "Clean":
        increase cleanliness
```

check_alive()

```
If hunger or happiness or
        discipline or cleanliness
        is 0:
        return False
else:
        return True
```

generate_mood()

```
mood == random from 0 to 3
If mood == 0:
        decrease hunger
Elif mood == 1:
        decrease happiness
Elif mood == 2:
        decrease discipline:
Elif mood == 3:
        decrease cleanliness
```

Tamagotchi

```
name: String
hunger: int
happiness: int
discipline: int
cleanliness: int
```

main()

```
Call create_new_tamagotchi()
```

Create_new_tamagotchi()

```
Input: name
my_tama = Tamagotchi(name)

alive = True
While alive:
        update my_tamas' status
        print my_tamas' status
        Ask user for action
        update my_tamas' status
        check if still alive
```

5. Start writing

- If you have a thorough architecture map with inputs and outputs specified you can easily divide the work between you.
- If you are working alone you can start in any corner of the project that you prefer.

Exercise

- Plan in groups a project that aims to code a calculator
  - 1. List all features and entities. Mark features as essential and non-essential given the small amount of time you have.
  - 2. Map the architecture of your program
  - 3. Add pseudo code
  - 4. Talk about how you could divide the work between you if you were to code this program.

Exercise – example solution

Essential

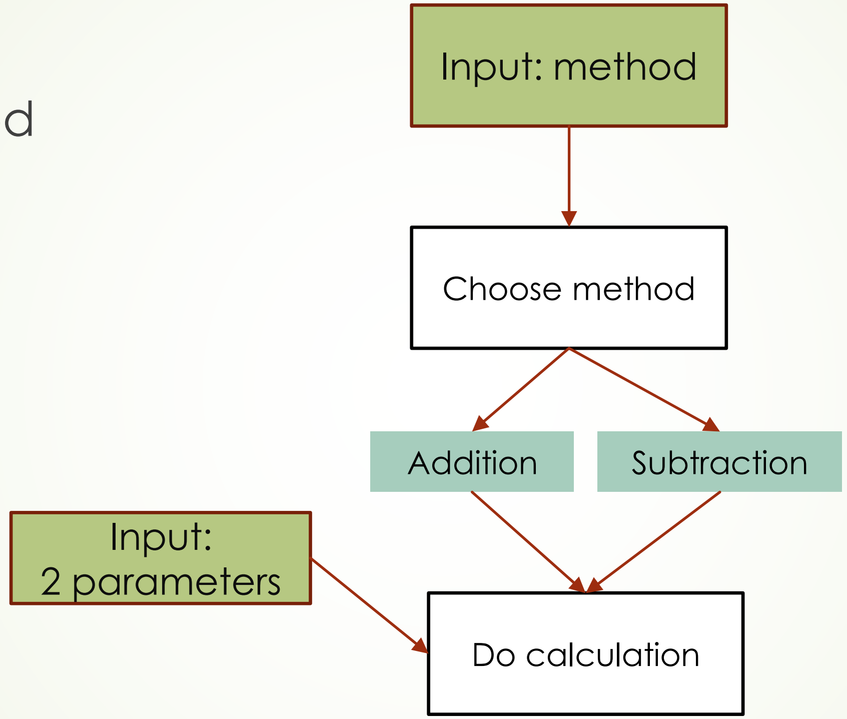
- Option to choose method
- 2 methods
    - Addition
    - Subtraction
- Do calculation

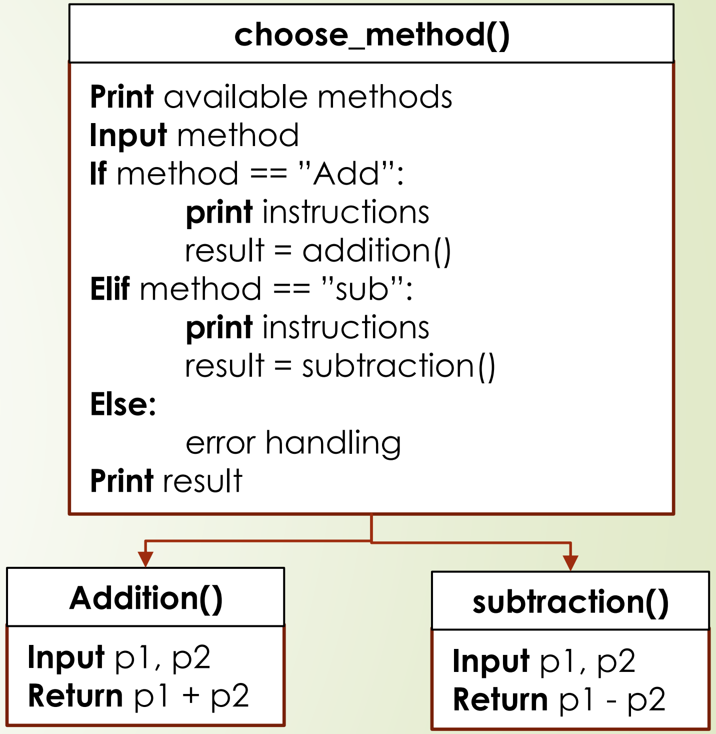
Non essential

- Multiplication
- Division
- Paranthesis
- Input more than 2 parameters

Entities

- 1 user





choose_method()

```
Print available methods
Input method
If method == "Add":
        print instructions
        result = addition()
Elif method == "sub":
        print instructions
        result = subtraction()
Else:
        error handling
Print result
```

Addition()

```
Input p1, p2
Return p1 + p2
```

subtraction()

```
Input p1, p2
Return p1 - p2
```

Suggested future discussion points

- UML diagrams
- Object oriented programming