Clean coder

Highlights

> Do and do not. There is no trying - Yoda

> A professional is confident in his abilities, and takes bold and calculated risks based on that confidence. A professional is not timid.

> It's the worst kind of unprofessional behavior to simply code from a spec without understanding why that spec makes sense to the business.

> The fact that bugs will certainly occur in your code does not mean you aren't responsible for them. The fact that the task to write perfect software is virtually impossible does not mean you aren't responsible for the imperfection

Professionalism

- It's all about taking responsibility
- It's about not creating bugs, which is impossible, but we must always try not to and if it happens: take responsibility
- Think about doctors: they take an oath to do no harm. In current society, critical processes run on code, why should programmers not do the same? Doctors do not want to re-open a body, lawyers do not want to re-open a case. We should not want to de-bug production code.
- You need to make sure your work actually addresses the needs of your employer: "its good to be passionate about what you do, but always keep an eye on the goals of the people who pay you. "
- 60 hours per week: 40 hours to do your job, 20 hours to study to be able to do the job in the future. These 20 hours should not be paid for: professionals spend time caring for their profession

Professional Coding

- Design code that is easy to test
- tests tests tests, many tests. About 3 chapters about this.
- Design code that is easy to change: if you make your code hard to change, to comprise the economics of your company.
- Don't write code when you are tired
- Don't believe in 'the zone' or 'coders flow': it might feel nice but the code you write when you are in the zone will bite you in the butt later.

Professional knowledge:

- Know all 24 patterns GOF
- Know the SOLID principles
- Know all you need to know about your domain: it is your responsibility to have all the context you need

Professional Communication:

- Don't try to explain your decisions too much: providing too much detail is an invitation for micro management.
- Don't say yes to everything, but do think in terms of solutions: be creative to find a way to help your employers/colleagues.

Expectation management

- Do not say things such as "I hope" or "we will try to".
- Make sure all stakeholders know the range of scenarios, and enforce them to create a fall-back plan.

Time management

- You will not get 20% more when you spend 20% time. Do overtime rarely.
- Sometimes you can not find data to settle a disagreement: flip a coin, and just go full-on for 1 of the sides. It is the best way to find out.

Life next to coding:

- creativity breeds creativity: reading fantasy books/looking at art etc will help you stay creative.

Management

- "Teams are harder to build than projects". So better to keep teams that can produce something start to end, and give them new assignments, then to create a new team for each assignment.