COMP 3550

4.3 — TECHNICAL DEBT: WHAT IT IS, WHERE IT COMES FROM

Week 4: Exceptional Testing & Technical Debt

TECHNICAL DEBT: THE METAPHOR

Like financial debt:

💸 You borrow [speed] today...

🕳️ But you pay interest [debugging, rewrites, and headaches] tomorrow

When you **have to** clean your room but really don’t want to.

TECHNICAL DEBT: THE METAPHOR

Like financial debt:

💸 You borrow [speed] today...

🕳️ But you pay interest [debugging, rewrites, and headaches] tomorrow

“We’ll hardcode this for now — we’ll refactor it later.”

- *B*ut “later” never comes without a plan.

COMMON CAUSES OF TECH DEBT

And not all are in your control...

Rushed features

COMMON CAUSES OF TECH DEBT

And not all are in your control...

Rushed features

Poor naming

COMMON CAUSES OF TECH DEBT

And not all are in your control...

Rushed features

Poor naming

BIG classes

COMMON CAUSES OF TECH DEBT

And not all are in your control...

Rushed features

Poor naming

BIG classes

Lack of Tests

COMMON CAUSES OF TECH DEBT

And not all are in your control...

Rushed features

Poor naming

BIG classes

Lack of Tests

Copy Paste Programming
Your code, your friends code,
stackoverflow, chatGPT

COMMON CAUSES OF TECH DEBT

And not all are in your control...

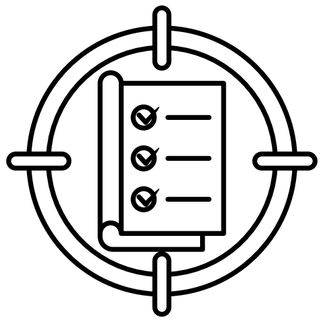
Rushed features

Poor naming

BIG classes

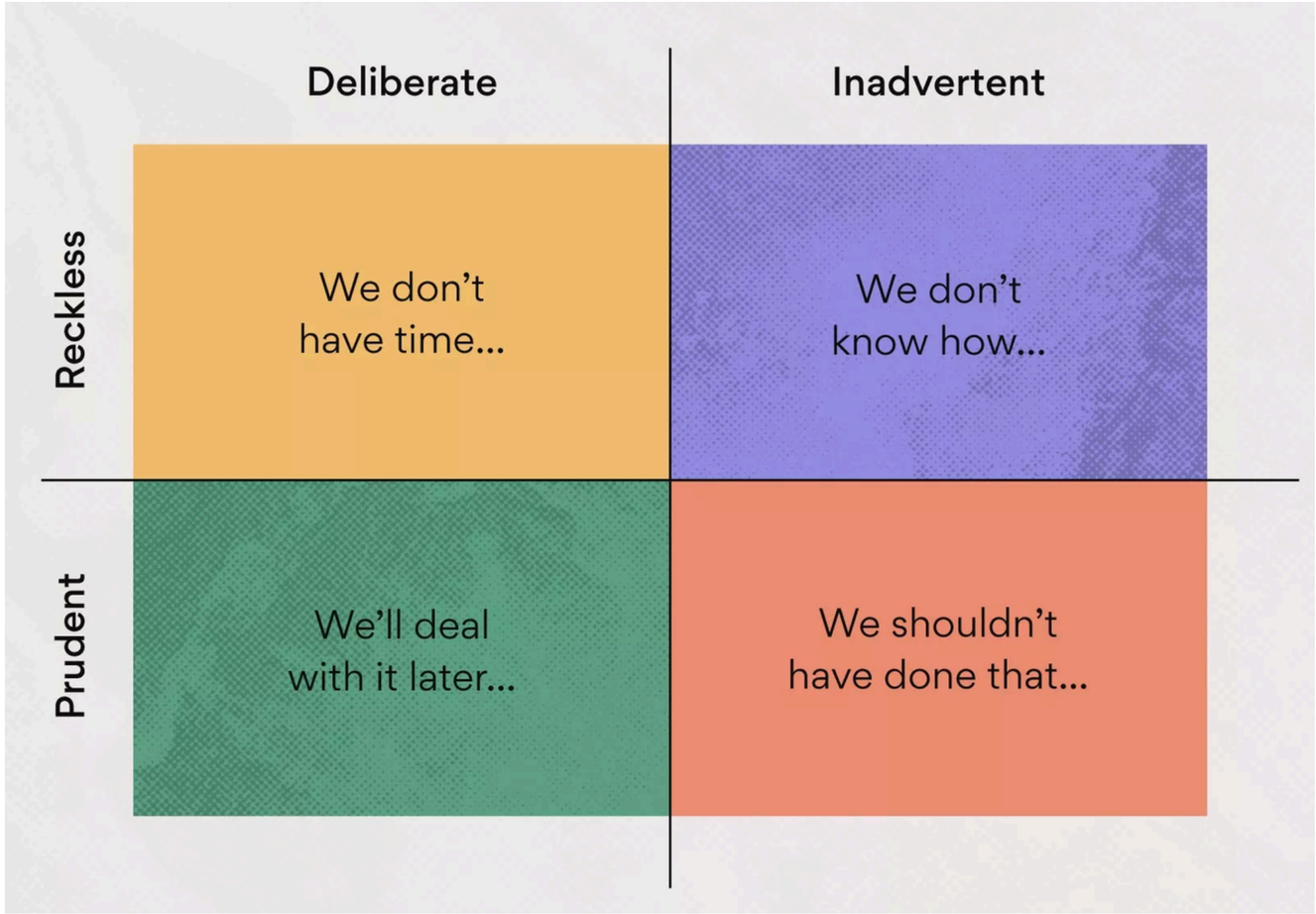
Lack of Tests

Copy Paste Programming
Your code, your friends code,
stackoverflow, chatGPT



Changing Requirements/Scope

TECH DEBT QUADRANT



Asana

CODE EXAMPLE

```
public void processOrder(Order order) {
    if (order == null || order.getItems().isEmpty()) {
        throw new IllegalArgumentException("Invalid order");
    }
    double total = 0;
    for (Item item : order.getItems()) {
        total += item.getPrice();
    }
    if (order.hasDiscount()) {
        total *= 0.9;
    }
    PaymentService ps = new PaymentService();
    ps.charge(order.getCustomer(), total);
    EmailService es = new EmailService();
    es.send(order.getCustomer().getEmail(), "Thanks!");
}
```

CODE EXAMPLE

```
public void processOrder(Order order) {
    validate(order);
    applyDiscount(order);
    chargeCustomer(order);
    sendConfirmationEmail(order);
}

private void validate(Order order) { ... }
private void applyDiscount(Order order) { ... }
private void chargeCustomer(Order order) { ... }
private void sendConfirmationEmail(Order order) { ... }
```

IDENTIFYING DEBT IN YOUR CODE

- Long methods
  - more than 10 lines?

- Inconsistent logic
  - Four different “validateUser” methods in four classes doing four different things

- Duplicated code
  - *if(user != null && !user.getName().trim().isEmpty()) { /* do work */ }*
  - null checks and .trim().isEmpty() on every textfield?
- TODOs and Questionable comments such as:
  - *// I know this is bad, will fix later*
  - *// shouldn’t hit this line of code ever*
  - *// not sure how else to catch this*

IDENTIFYING IT AS YOU CODE

- A quick digression on cognitive inertia
  - We need some rules that help us to interupt our default train of thought

IDENTIFYING IT AS YOU CODE

- A quick digression on cognitive inertia
  - We need some rules that help us to interupt our default train of thought
- When designing code:
  - What are you trying to accomplish? Be wary of ‘ands’
- The dreaded Copy/Paste
  - any copy/paste should give you **pause**
- During testings
  - Unit testing, what does this method/class/etc do? Is it well defined?

We will learn some acronyms soon enough to help with this as well

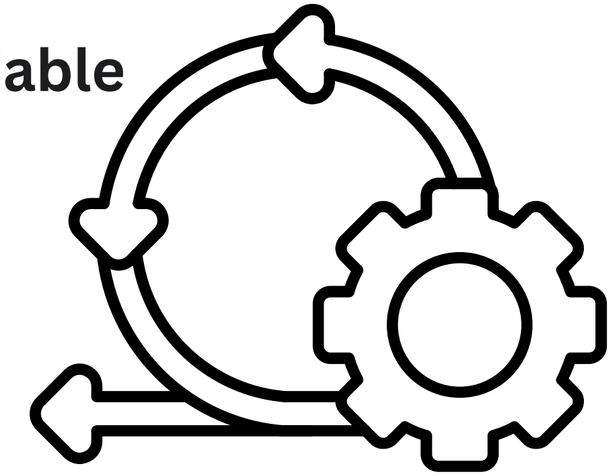
PAYING IT DOWN

- Prioritize and Document
  - "Tech Debt Log" (shared doc)
- Refactor incrementally
  - Sprint Planning
- Add tests first
  - helps us to make sure we don’t break anything

TECH DEBT ≠ BAD TEAM

- Tech Debt is normal and **not completely avoidable**

- Interrupt the cognitive inertia
- Notice it
- Document
- Help Each Other



PAUSE AND PROJECT REFLECT

Find one piece of tech debt in your code.
Create a task or issue for cleaning it up later.

Now do the same for a piece of code you did not write in your project (a teammates code). Remember to be kind, we all create debt, it’s about working together and growing.