COMP2511

WEEK 10

Did you get much sleep?

A G E N D A

- myExperience
- Finding Patterns
- Code and Design Smells
- Serverless Archiecture

MYEXPERIENCE

Please complete your myExperience survey of your experience of me!

It will greatly help me improve for whatever I am teaching in the future but also help the course improve for future iterations.

All constructive feedback is welcome. I would like to become a better tutor for future students!

Finding Patterns

FINDING PATTERNS

Sorting collections of records in different orders.

FINDING PATTERNS

Strategy Pattern

This what Java does with the Collections.sort() method. A Comparator can be provided to determine the order in which elements are sorted.

FINDING PATTERNS

Listing the contents of a file system.

FINDING PATTERNS

Composite pattern

Both folders and files are filesystem entries. Files form the leaves, folders can contain files or other folders.

FINDING PATTERNS

Updating a UI component when the state of a program changes.

FINDING PATTERNS

Observer pattern

If the state of the program is the subject and the UI an observer, the UI will be notified of any changes to the state and update accordingly.

FINDING PATTERNS

A frozen yogurt shop model which alters the cost and weight of a bowl of frozen yogurt based on the toppings that customers choose to add before checkout.

FINDING PATTERNS

Decorator pattern

Toppings (new behaviours) are added dynamically at runtime by users (customers).

FINDING PATTERNS

You are making a stealth game and want the enemy guards to be alerted when the player produces a loud noise.

FINDING PATTERNS

Observer pattern

The player's noise-making actions are the subject, and enemy guards are observers that get notified when loud sounds occur within their detection range.

FINDING PATTERNS

Creating a cross-platform application that needs to generate different types of UI components (buttons, menus, dialogs) that match the look and feel of each operating system (Windows, macOS, Linux).

FINDING PATTERNS

Abstract Factory pattern

Each platform (Windows, macOS, Linux) has its own concrete factory that creates families of related UI components with consistent styling for that platform.

FINDING PATTERNS

A text editor application where you can apply multiple formatting options (bold, italic, underline, highlight, different fonts) to text, with the ability to combine these effects and add or remove them dynamically.

FINDING PATTERNS

Decorator pattern

Each formatting option (bold, italic, underline, etc.) acts as a decorator that wraps the text component, allowing multiple formatting effects to be layered and combined flexibly.

Code and Design Smells

CODE AND DESIGN SMELLS

Mark, Bill and Jeff are working on a PetShop application. The PetShop has functionality to feed, clean and exercise different types of animals. Mark notices that each time he adds a new species of animal to his system, he also has to rewrite all the methods in the PetShop so it can take care of the new animal.

CODE AND DESIGN SMELLS

Mark, Bill and Jeff are working on a PetShop application. The PetShop has functionality to feed, clean and exercise different types of animals. Mark notices that each time he adds a new species of animal to his system, he also has to rewrite all the methods in the PetShop so it can take care of the new animal.

Code smell - Divergent change
Design problem - Open Closed Principle, high coupling

CODE AND DESIGN SMELLS

```java
public class Person {
    private String firstName;
    private String lastName;
    private int age;
    private int birthDay;
    private int birthMonth;
    private int birthYear;
    private String streetAddress;
    private String suburb;
    private String city;
    private String country;
    private int postcode;

    public Person(String firstName, String lastName, int age, int birthDay,
                  int birthMonth, int birthYear, String streetAddress, String suburb,
                  String city, String country, int postcode) {
        this.firstName = firstName;
        this.lastName = lastName;
        this.age = age;
        this.birthDay = birthDay;
        this.birthMonth = birthMonth;
        this.birthYear = birthYear;
        this.streetAddress = streetAddress;
        this.suburb = suburb;
        this.city = city;
        this.country = country;
        this.postcode = postcode;
    }
    // Some various methods below
    // ....
}
```

CODE AND DESIGN SMELLS

Data clumps, long parameter list

Refactor by making more classes for birthday and address ("Extract Class"/ "Introduce Parameter Object")

Design problem - DRY and KISS

CODE AND DESIGN SMELLS

```
public class MathLibrary {
    List<Book> books;

    int sumTitles {
        int total = 0
        for (Book b : books) {
            total += b.title.titleLength;
        }
        return total;
    }
}

public class Book {
    Title title; // Our system just models books as titles (content doesn't matter)
}

public class Title {
    int titleLength;

    int getTitleLength() {
        return titleLength;
    }

    void setTitleLength(int tL) {
        titleLength = tL;
    }
}
```

CODE AND DESIGN SMELLS

Inappropriate intimacy (accessing public fields)

Message chains

Data classes/Lazy classes Design smell - High coupling, from encapsulation being broken

Fixes - make things private, just delete the classes and represent titles as strings

CODE AND DESIGN SMELLS

1. How do these code smells cause problems when developing code?
    - Reusability, Maintainability, Extensibility
    - Second and third examples are opposite problems (not enough classes vs too many classes) - you can take any refactoring too far
2. Is a code smell always emblematic of a design problem?
    - No - e.g "switch statements" and "comments" are often listed as code smells but are not always actually smells

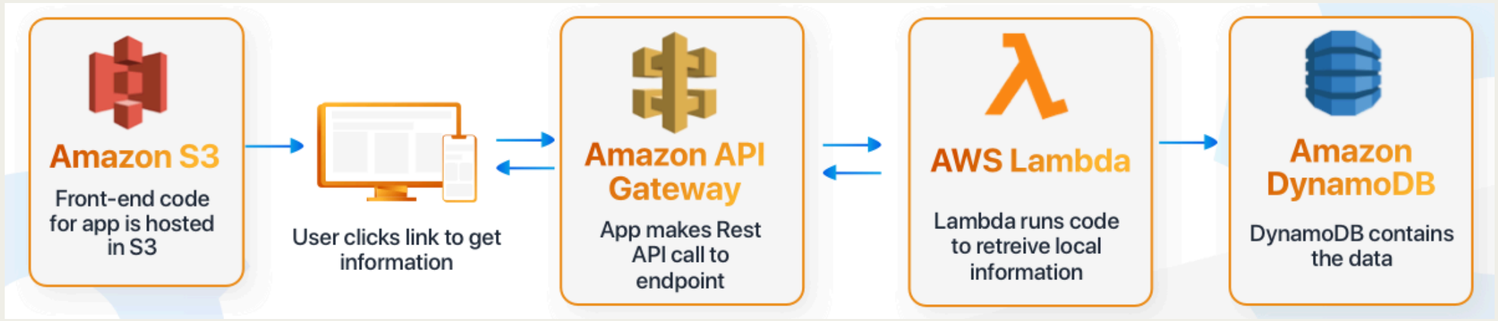
Serverless Architecture

Content of slides inspired by Rebecca Hsu’s (COMP2511 Tutor) Slides.

SERVERLESS

Allows developers to build and run applications without managing infrastructure

- Apps are built as a **collection of functions** deployed to a **cloud provider** who automatically provision, scale and manage the infrastructure
- Code is deployed as **small, stateless functions (FaaS)** which are triggered by **events** (e.g. HTTP request, file upload)
- Developers only need to worry about writing code and are billed for execution time to the millisecond.



SERVERLESS

Strengths	Challenges
• High scalability and elasticity: Cloud providers **adjust resources** based on **real-time demand** (i.e. instant scaling during traffic surges) • **High availability** due to multi-zone distribution, **eliminates maintenance** • **Cost optimisation**: Billing based off execution time • **Faster deployment**/faster **time-to-mark**: Abstracting away infrastructure management, **reduced operational overhead**. Independent function updates without redeploying the entire app.	• Cold starts: **Takes time to initiate internal resources** for the first function request • Limited control, **reliant on the provider** and their **ecosystem** • **Debugging complexity:** Functions run in short-lived instances without persistent state, making it difficult to replicate problems and harder to track request flows across functions

SERVERLESS

Describe a scenario where a serverless function would be used to handle user-submitted form data.

SERVERLESS

Describe a scenario where a serverless function would be used to handle user-submitted form data.

- When a user submits a form on a website (e.g. a contact or feedback form), this action can trigger a serverless function via an API Gateway. The function could then validate the data and store it in a database.

- Serverless is ideal here because we only want it to run when the form is submitted. We don't know how consistent traffic will be (nor how much traffic we'll get). This reduces the cost and removes the need to manage a backend server for infrequent submissions.

SERVERLESS

In what way can serverless functions improve scalability for an e-commerce site during high traffic periods like Black Friday?

SERVERLESS

In what way can serverless functions improve scalability for an e-commerce site during high traffic periods like Black Friday?

- Serverless functions automatically scale to handle a large number of concurrent requests. E.g each time a user places an order or views a product, a function can be triggered to update inventory or process payments.

- During high traffic events like Black Friday, this elasticity ensures smooth performance without provisioning infrastructure ahead of time, and you only pay for actual usage.

KAHOOOTTTTTTTTTT!!!!!!

LABBBBBBB

SAMPLE EXAM