Welcome!

- We'll start in a moment :)
- We may record tonight's event and plan to take screenshots for social media.
    - ***If you want to remain anonymous***, use your first name & keep video off.
- We'll introduce the hosts and break in-between for Q/A.
- We will make some time for Q&A at the end of the presentation as well.
- Online event best practices:
    - Mute yourself when you aren't talking.
    - Turn on your video if you feel comfortable!

WOMEN WHO CODE

- Welcome from WWCode!
- Our mission: Inspiring women to excel in technology careers.
- Our vision: A world where women are representative as technical executives, founders, VCs, board members and software engineers.

Prachi Shah

Senior Software Engineer | Metromile

WWCode Digital + **Backend**
Backend Study Group

April 22, 2021

WOMEN WHO CODE

Resources

- Third ever Backend Study Group session!
- **Topic: Software Design Patterns [Part 1 of 5]**
  - Creational Design Patterns
  - Structural Design Patterns
  - Behavioral Design Patterns
  - Anti-patterns and design principles
  - Interview Questions
- WWCode GitHub and Demo
- WWCode YouTube channel:
  - March 25, 2021 session recording: Backend Study Group session 1
  - April 8, 2021 session recording: Backend Study Group session 2
- Technical Tracks
- Check our Digital Events
- Get updates – join the Digital mailing list!
- Survey

Agenda

- What is Backend Engineering?
- Software Design
- Object Oriented Programming (OOP) principles
- Design patterns
- Types of patterns
- Top 3 Creational patterns
- Q/A
- Resources:
  - Software design pattern
  - Design Patterns in Java
  - Design patterns
  - Design Patterns
  - Head First Design Patterns book

Backend Engineering

- What is Backend Engineering?
- Design, build and maintain server-side web applications.
- Concepts: Client-server architecture, API, micro-service, database engineering, etc.

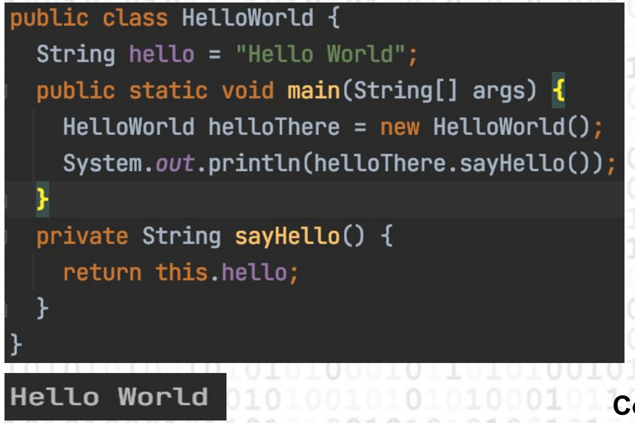
Software Design

- Defining the architecture, modules, interfaces and data
- Solve a problem or build a product
- Define the input, output, business rules, data schema
- Design patterns solve common problems
- 3 Types:
    - UI design: Data visualization and presentation
    - Data design: Data representation and storage
    - Process design: Validation, manipulation and storage of data
- Distributed systems, storage, performance, deployment, availability, monitoring

Backend Engineering

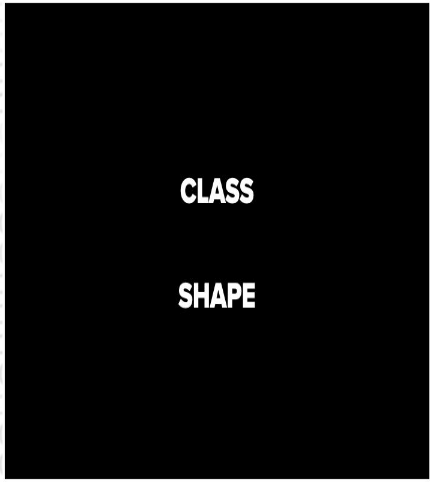
Object Oriented Programming (OOP)

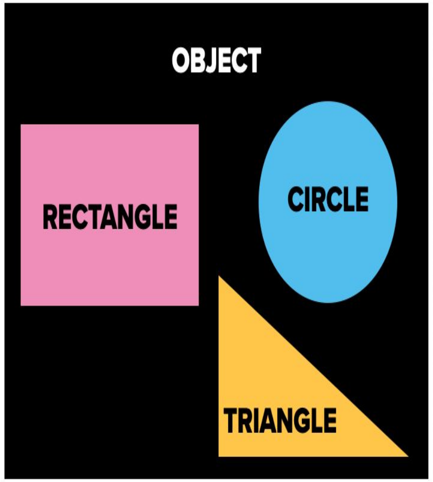
- Create objects with data (attributes) and functions (methods)
- Structured and minimal code
- Don’t Repeat Yourself [DRY] to easily change, maintain and debug
- Class: Template: data and functions
- Object:
    - Instance of a class
    - Many objects per class



```
public class HelloWorld {
  String hello = "Hello World";
  public static void main(String[] args) {
    HelloWorld helloThere = new HelloWorld();
    System.out.println(helloThere.sayHello());
  }
  private String sayHello() {
    return this.hello;
  }
}
```

```
Hello World
```



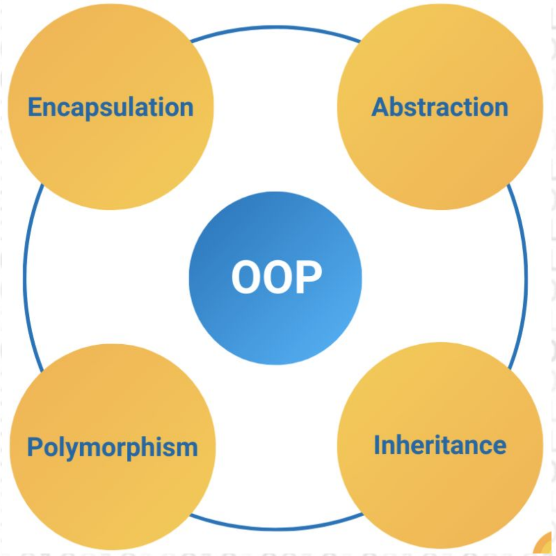


Backend Engineering

- **Constructor**:
    - Initialize objects: initialize attributes
    - Called when objects are created
    - Name matches class name, has no return type, default and custom constructors

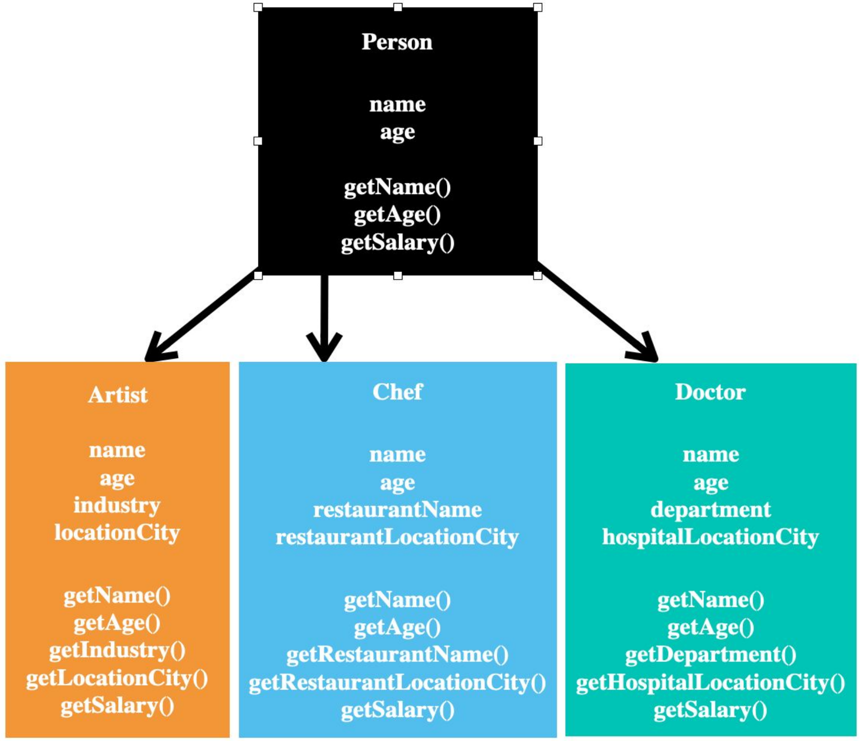
- **Access Modifiers**:
    - private: visible inside the same class
    - public: visible everywhere
    - protected: same class, subclasses, packages

- **Principles**:
    - Abstraction
    - Encapsulation
    - Polymorphism
    - Inheritance



Backend Engineering

- **Abstraction**:
  - Hide complexity and details
  - Caller does not need to know the details
  - Each class has its own abstraction
  - Easy to maintain code and add features
  - Example: Instant Pot
  - Types:
    - Data abstraction:
      - Getter and setter methods
      - Example: getName(), getAge()
    - Process abstraction:
      - Function implementation varies
      - getSalary()
  - Code example



WOMEN WHO CODE

Backend Engineering

- **Encapsulation**:
  - Bundle data and methods into one unit (class)
  - Entity: data and operations match real-world scenario
  - Hide data from users
  - Declare attributes as **private**
  - **get()** and **set()** methods to access data
  - Better control over data access and methods
  - Improved security of data
  - Data can be read-only or write-only
  - Code example

Chef

name
age
hours
rate
restaurant

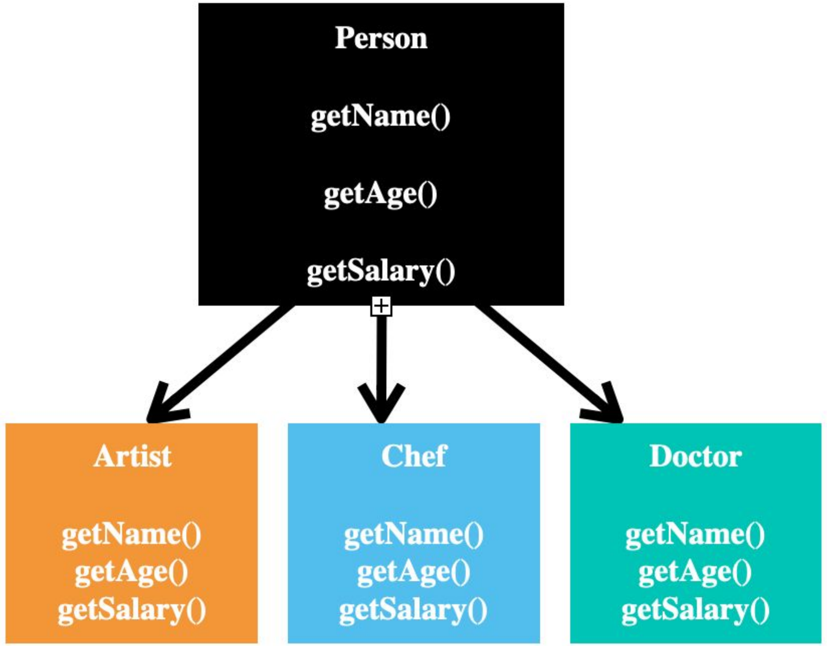
getName()
getAge()
getHours()
getRate()
getRestaurant()

setName(name)
setAge(age)
setHours(hours)
setRate(rate)
setRestaurant(restaurant)

calculateSalary(hours, rate)
hours * rate

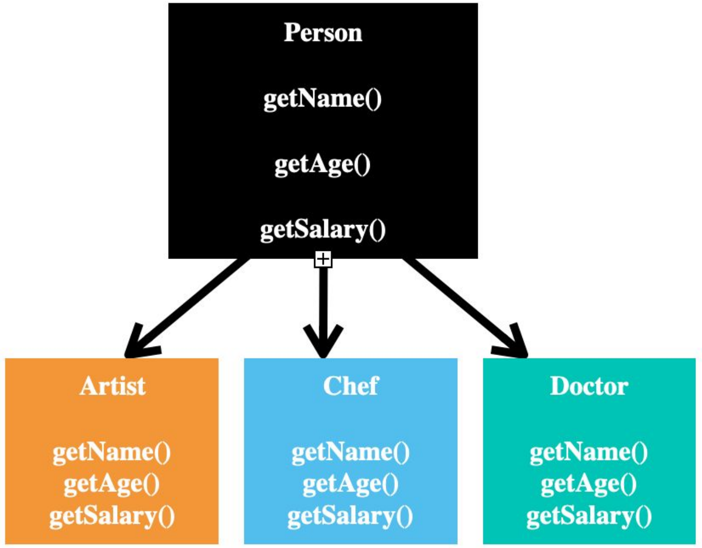
Backend Engineering

- **Inheritance**:
  - Derive a class from another class
  - Classes share attributes & methods
  - Hierarchy of super class, sub class
  - `extends` keyword
  - Reusability
  - Code example



Backend Engineering

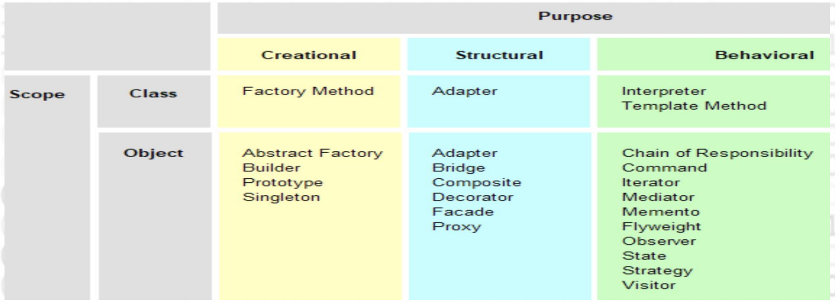
- **Polymorphism**:
    - Use same interface for different classes
    - `is-a` relationship
    - `implements` keyword
    - Interface has public methods without implementation
    - Implementing class overrides all of these methods
    - Implementing class provides own function/ logic
    - Code example



Backend Engineering

Design Patterns

- Set of template solutions that can be reused
- Shared pattern vocabulary
- Improved code maintainability, reusability and scaling
- Not a library or framework, but recommendations for code structuring and problem solving
- Adapt a pattern and improve upon it to fit application needs
- Leverages OOP for flexible and maintainable designs
- Defines relationship between objects, loosely coupled objects, secure objects

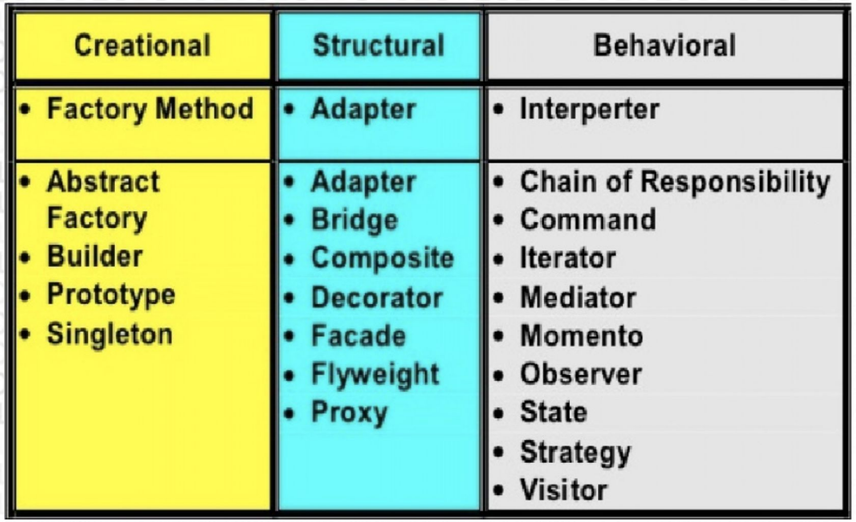


		Purpose		
		Creational	Structural	Behavioral
Scope	Class	Factory Method	Adapter	Interpreter Template Method
	Object	Abstract Factory Builder Prototype Singleton	Adapter Bridge Composite Decorator Facade Proxy	Chain of Responsibility Command Iterator Mediator Memento Flyweight Observer State Strategy Visitor

Backend Engineering

Types of Design Patterns

- **Creational**:
  - Initialize a class and instantiate the objects
  - Decoupled from implementing system
  - Singleton, Factory, Builder
  - Abstract Factory, Prototype
- **Structural**:
  - Class structure and composition
  - Increase code reusability and functionality
  - Create large objects relationships
  - Adapter, Facade, Decorator, etc.
- **Behavioural**:
  - Relationship and communication between different classes
  - Observer, Strategy, Iterator, etc.



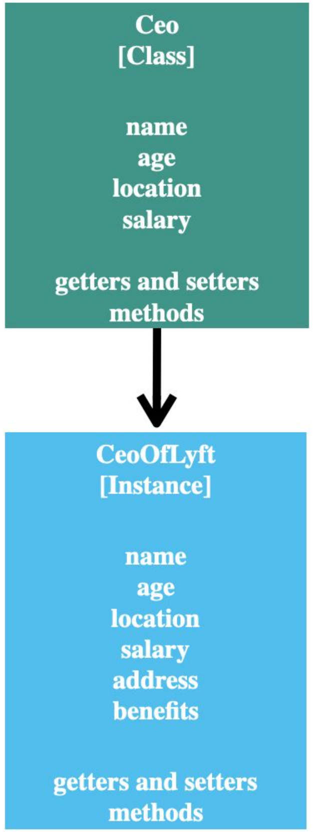
Creational	Structural	Behavioral
• Factory Method	• Adapter	• Interperter
• Abstract Factory • Builder • Prototype • Singleton	• Adapter • Bridge • Composite • Decorator • Facade • Flyweight • Proxy	• Chain of Responsibility • Command • Iterator • Mediator • Momento • Observer • State • Strategy • Visitor

WOMEN WHO CODE

Backend Engineering

Creational Design Patterns:

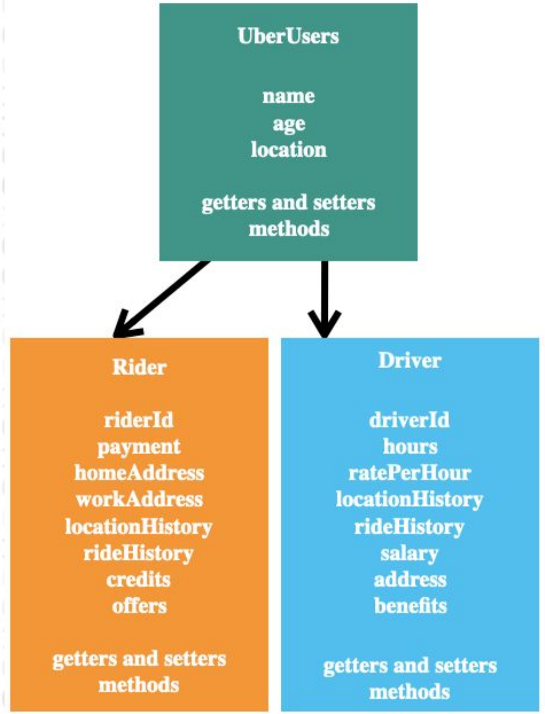
- **Singleton:**
  - One instance only
  - Instance variable is `static`
  - `private` constructor
  - Caller gets the instance from a `getInstance()`
  - Lazy initialization: Instance is created and initialized on-demand
  - Eager initialization: Instance is created and initialized on class load
  - One instance per singleton per Java Virtual Machine (JVM)
  - Example: Company has one CEO; University has one Proctor
  - Example: Log4j logging program
  - Code: `Ceo ceoOfLyft = Ceo.INSTANCE;`
  - Code example

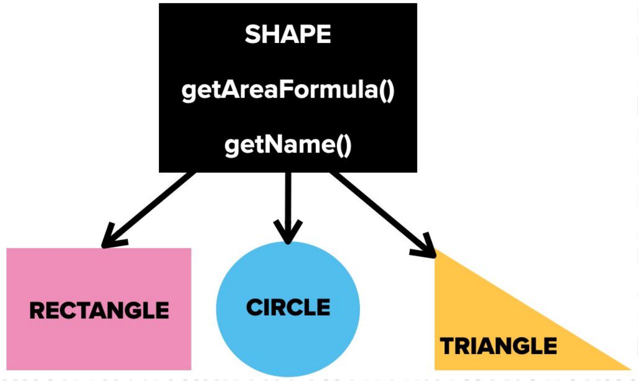


Backend Engineering

Factory:

- Create an object by hiding the creation logic
- Use a common `interface` to create objects
- Create a new instance on-demand and initializes fields
- Reduces code duplication, provides consistent behavior
- Easy to maintain classes as creation is centralized
- Loosely coupled classes
- Example: Uber users
- Code example

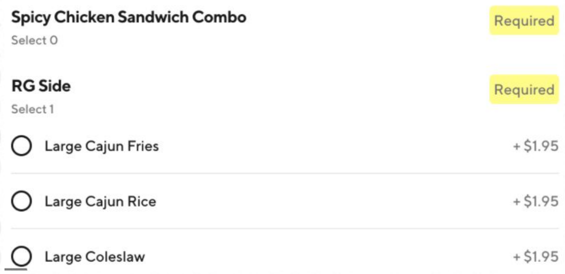




Backend Engineering

Builder:

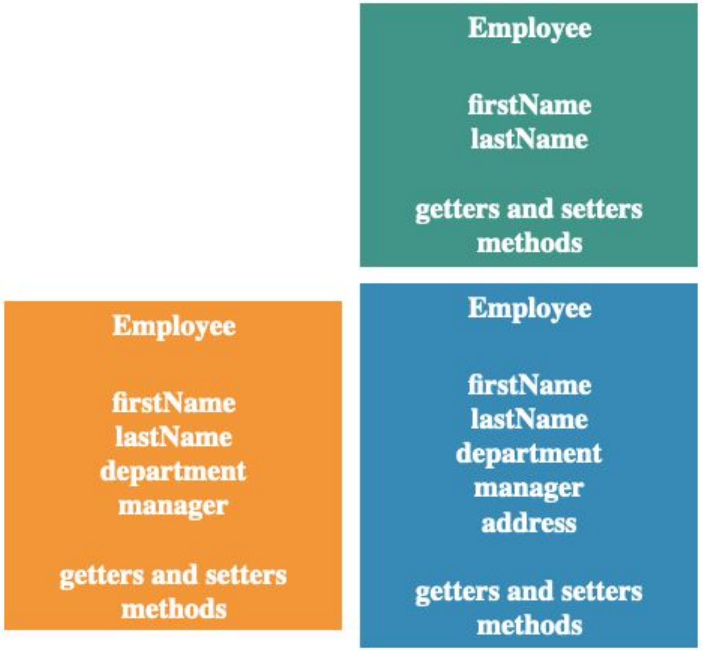
- Build custom objects of a class
- Objects can be different
- Use the same creation logic
- Seperate the construction and representation
- Flexible design, readable code, complete objects
- Example: Ordering food from DoorDash
- Code example



Spicy Chicken Sandwich Combo — Required
Select 0

RG Side — Required
Select 1

○ Large Cajun Fries	+ $1.95
○ Large Cajun Rice	+ $1.95
○ Large Coleslaw	+ $1.95



WOMEN WHO CODE

Backend Study Group

WOMEN WHO CODE