# Agenda

- **What is CRC**
- Why use CRC
- Example
- Control style
- Exercise
- Stereotypes
- Summary

# What is CRC

- A tool for designing modules and applications
- Short for
  - Candidate
  - Responsibilities
  - Collaborators

# What is CRC

| **Candidate**:<br>AccountService | **Collaborators**:<br>● FundsTransferer<br>● AccountManager |
| --- | --- |
| **Responsibilities**:<br>● Transfers funds between accounts<br>● Creates account<br>● Closes account | |

# Agenda

- What is CRC
- **Why use CRC**
- Example
- Control style
- Exercise
- Stereotypes
- Summary

# Why use CRC

- We already do it one way or another
- It’s a way of expressing design
- Helps reasoning and communication

# Agenda

- What is CRC
- Why use CRC
- **Example**
- Control style
- Exercise
- Stereotypes
- Summary

# Example

- Water boiler

| Candidate | Collaborators |
| --- | --- |
| Responsibilities | |

# Example

| Water boiler | Heater<br>Reservoir<br>Switch |
| --- | --- |
| Boils water | |

| Heater |
| --- |
| Heats water |

| Reservoir |
| --- |
| Holds water |

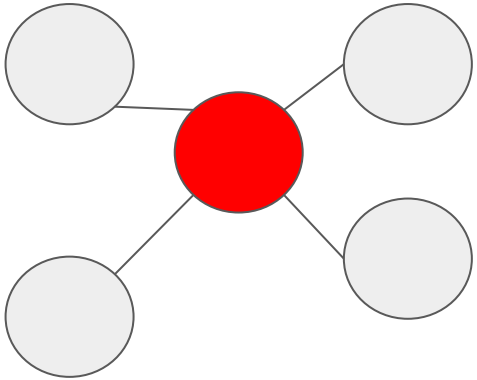
| Switch | Heater |
| --- | --- |
| Provides electricity | |

# Agenda

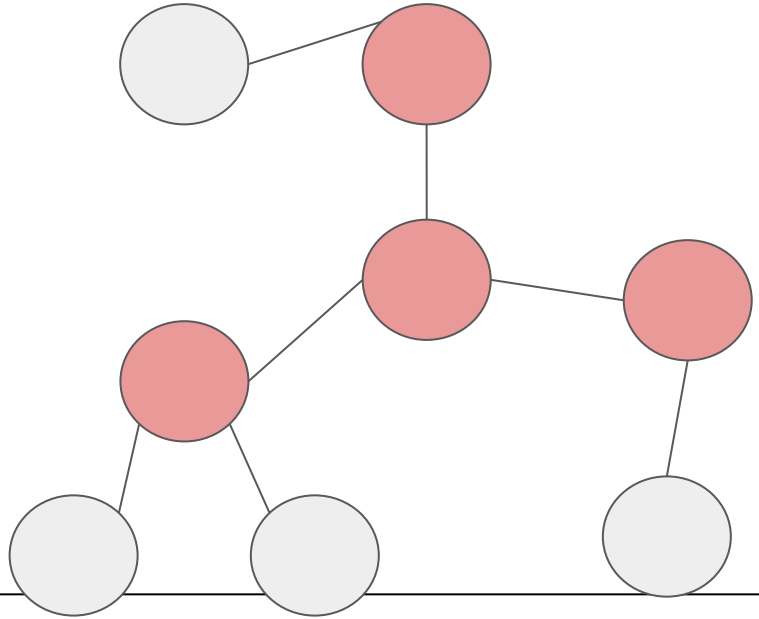
- What is CRC
- Why use CRC
- Example
- **Control style**
- Exercise
- Stereotypes
- Summary

# Control style

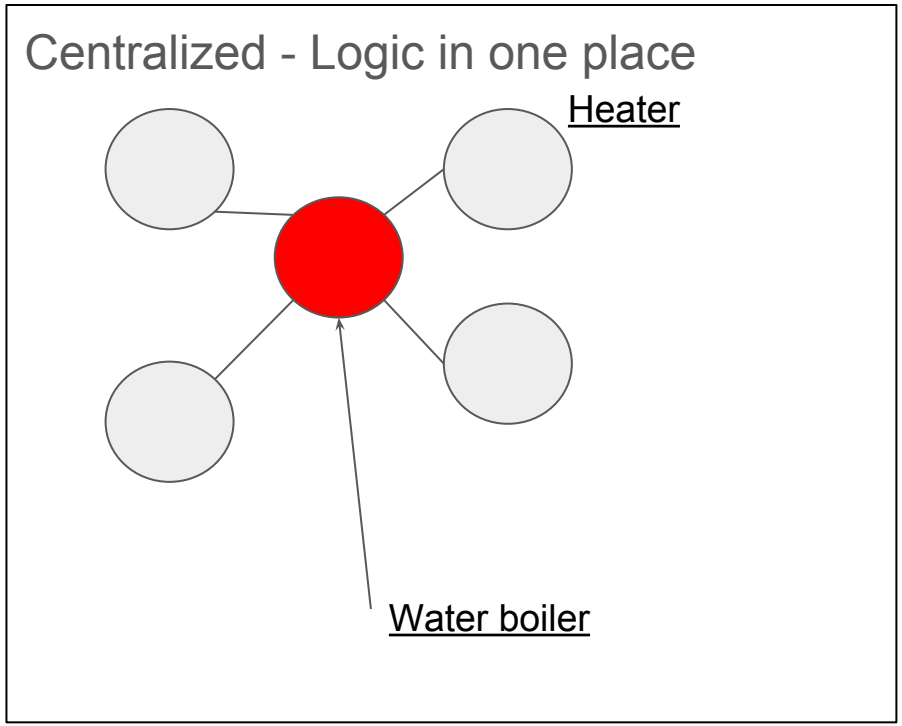
Centralized - Logic in one place

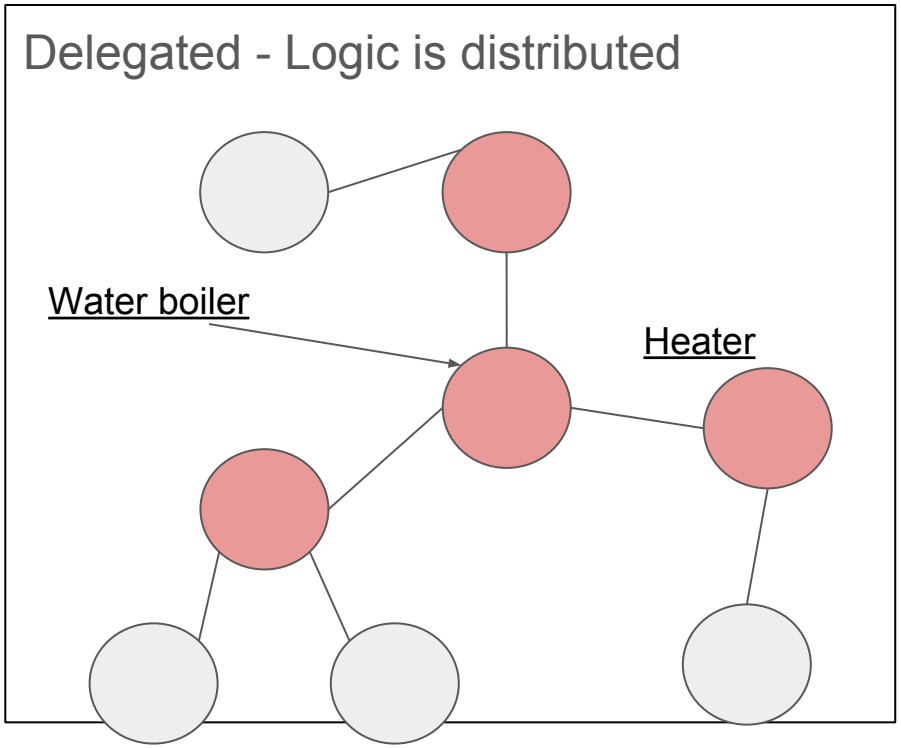


Delegated - Logic is distributed



# Control style



# Control style

## Centralized - Logic in one place

Pros:

- Application logic is in one place

Cons:

- Control logic can get overly complex

## Delegated - Logic is distributed

Pros:

- Fewer dependencies
  - Easier to change as changes typically affect fewer objects
  - Easier to understand

Cons:

- Too much distribution of responsibility can lead to weak objects and weak collaborations

# Agenda

- What is CRC
- Why use CRC
- Example
- Control style
- **Exercise**
- Stereotypes
- Summary

# Exercise

- Bicycle

# Exercise

- Analyzing a class
  - What are its responsibilities?
  - Who are its collaborators?

# Agenda

- What is CRC
- Why use CRC
- Example
- Control style
- Exercise
- **Stereotypes**
- Summary

# Stereotypes

- Service provider
- Information holder
- Coordinator
- Controller
- Interfacer
- Structurer

# Service provider

*Performs work*

| AccountValidator | Account |
| --- | --- |
| Validates that account has positive balance | |

# Information holder

*Knows things*

| Account | |
| --- | --- |
| Knows balance and account number | |

# Coordinator

*Coordinates actions*

| AccountService | FundsTransferer<br>AccountManager |
| --- | --- |
| Transfers funds<br>Creates account<br>Closes account | |

# Controller

*Makes decisions*

| AccountManager | Account<br>AccountValidator |
| --- | --- |
| Creates account<br>Closes account | |

# Interfacer

*Transforms information*

| AccountRepository | Account<br>JDBC |
| --- | --- |
| Persists account information | |

# Structurer

*Maintains relationships*

| Loan | Customer<br>Account<br>Debt |
| --- | --- |
| Maintains relationship between customer, account and debt | |

# Summary

- How to use CRC to design modules
- Pros and cons with centralized and delegated control style
- Analyzing existing modules