Software Architecture Intro

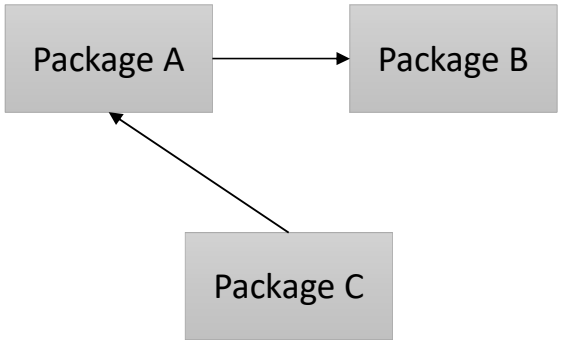
CSCI 3300/5300

Design & Architecture

- Are these concepts the same or different?
- Common definitions:
  - Architecture – high level design of the system
  - Design – low level details
- Problem with this definition: high level design depends on the low level details
- Design and Architecture have a strong connection
- Alternate definition:
  - Design – definition of the system modules and interactions between modules
  - Architecture – bundling modules into components/packages and interactions between components

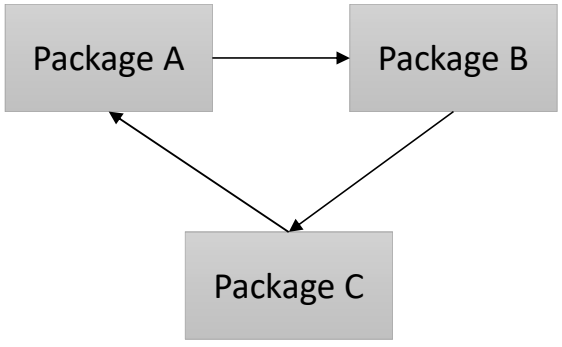
The Acyclic Dependencies Principle

- Suppose a new version of package B is released
- Developers of package A can choose to use the new version or not
- Developers of package C can choose to use the new version of packages A & B or not



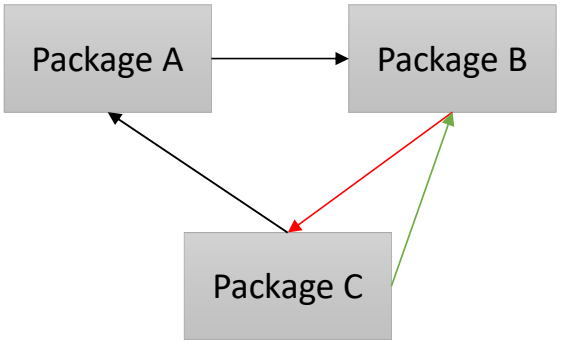
The Acyclic Dependencies Principle

- Suppose a new version of package B is released
- Changes to package B require changes to package C, which require changes to package A.
- A, B, & C now became one large package



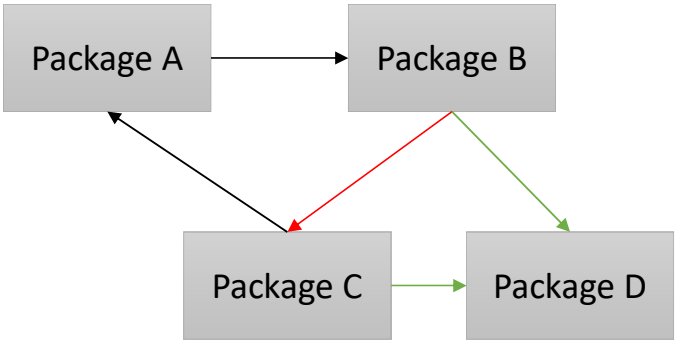
Breaking the cycle: Dependency Inversion Principle

- There is a class X in package C that package B needs.
- Create an "interface" (in Java terms) for that class, put it in package B and have class X implement that interface
- Now package C depends on package B



Breaking the cycle: Jitters

- Component structure is changing as requirements change
- Introduce a new component, on which both B and C depend



Family Tree Project

- https://github.com/kate-holdener/family-tree
- Overview

Monolithic "architecture"

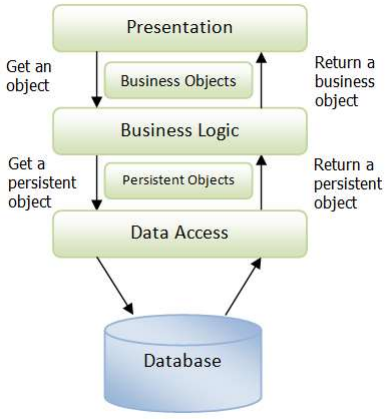
- Everything in one package
- Four classes:
  - FamilyMember – one person with mother and faather
  - FamilyTree – a collection of family members
    - Loads data from csv file
    - Identifies parents, siblings, and children of a given person
  - FamilyTreeGui – Java SWING API for the application
  - BuildFamilyTree – command line interface for the application

Evaluate this architecture in terms of

- Agility – if requirements change, how easy is it to change the application?
- Easy of deployment – if one component changes, how easy is it to re-deploy the software
- Testability – how easy is it to test this solution
- Scalability – as the application grows, how easy it is to maintain the code, find the right package, etc
- Ease of development – can a team of developers easily split up the work without interfering with each other?

Layered Architecture

- Several "horizontal" layers
- Each layer represents a different "user"
- Each layer can use only adjacent layers
- In our family tree example
  - User interface
  - Application
  - Database
  - (Driver)



Evaluate this architecture in terms of

- Agility – if requirements change, how easy is it to change the application?
- Easy of deployment – if one component changes, how easy is it to re-deploy the software
- Testability – how easy is it to test this solution
- Scalability – as the application grows, how easy it is to maintain the code, find the right package, etc
- Ease of development – can a team of developers easily split up the work without interfering with each other?

Useful links:

- https://www.oreilly.com/ideas/software-architecture-patterns/page/3/event-driven-architecture
- https://www.oreilly.com/ideas/software-architecture-patterns/page/7/pattern-analysis-summary