CSCI E-97
Lecture 9

October 30, 2014

Outline

- Announcements
- Assignment 2 feedback
- Persistence and Object-Relational-Mapping
- Assignment 4

Announcements

- Assignment 3 is due tonight (Thursday) at 11:59pm EST.
- Assignment 4 will be handed out and discussed in the second half of the lecture
  - Authentication Service for Mobile Application Store
  - Due November 20th

Assignment 2 Results

- Designs were generally very good
- Feedback indicated that peer design reviews were helpful and helped get design done early
- In most cases the thought process that went into the designs helped streamline the implementation

- Creating the design gets easier with practice
- We are learning: new concepts, tools, and ways of working
- So it takes time to learn and becomes easier with practice

- Excellent questions and discussion on forum
- It is the job of the designer to get clarity on requirements from the product manager and/or customer

Software Design Goal

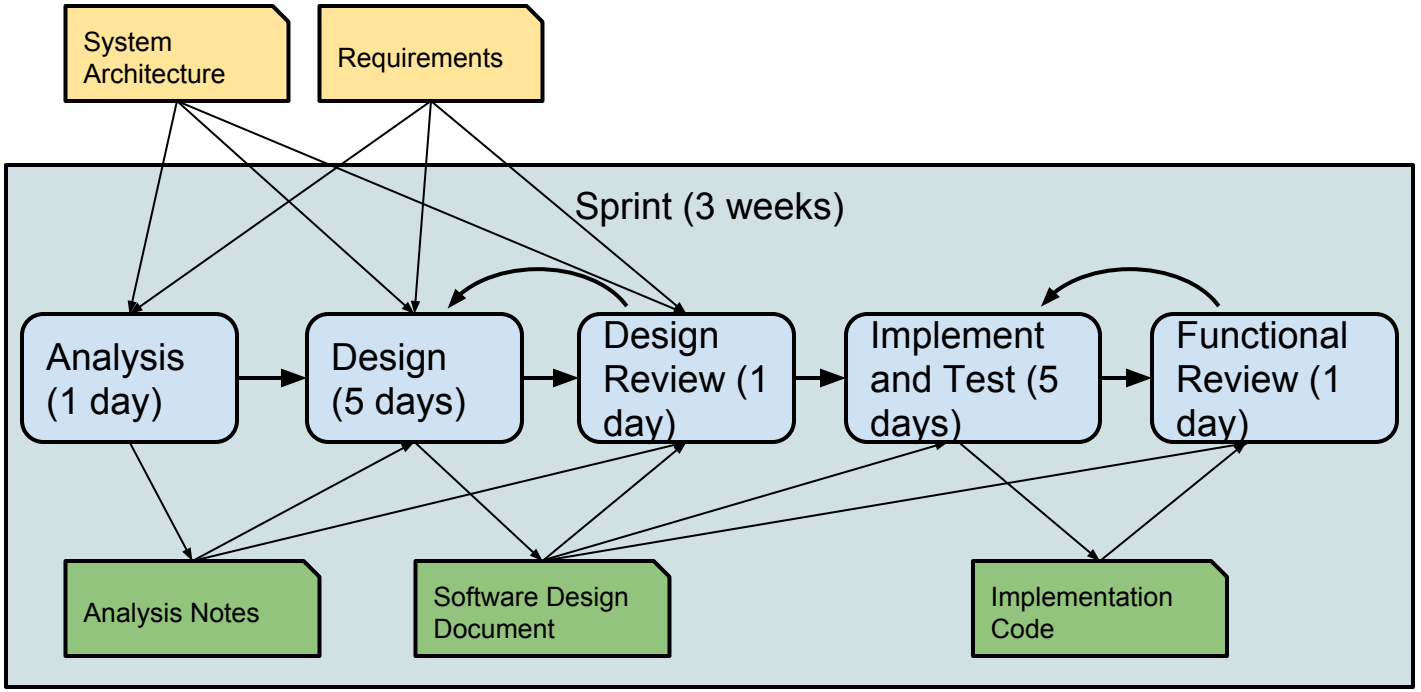
Ultimate goal of software design is to increase the quality of our software

- Collect and understand the requirements
- Analyse the problem
  - Considering different ways to solve
  - Selecting an approach
- Document in a design
  - Apply design principles
  - Object Modeling
  - UML diagrams
- Review the design
- Implement the design (with tests)

Especially helpful in a team environment or when multiple teams are working together

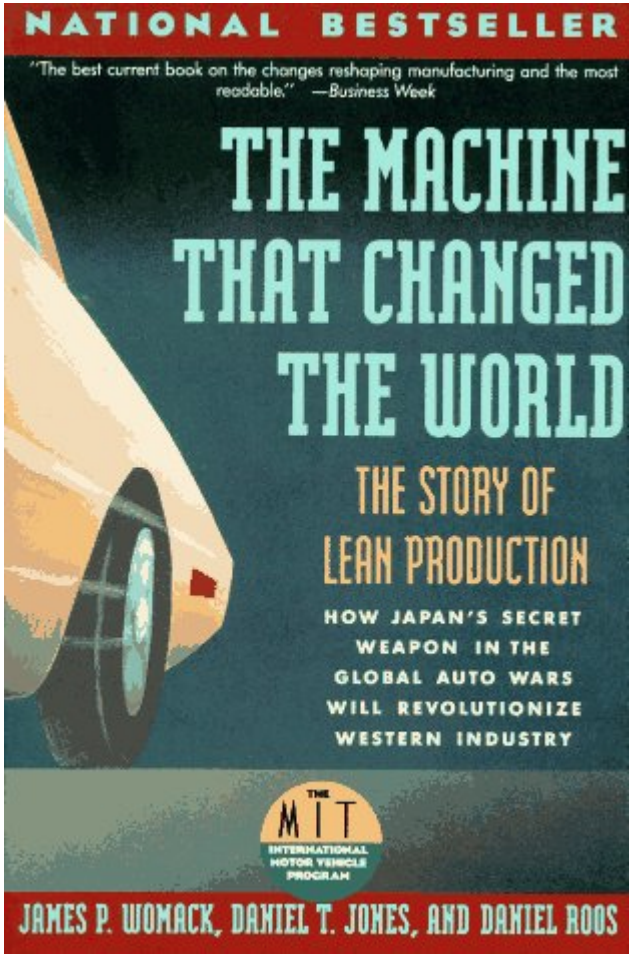
Agile Software Development Process

Our Sprint based software development process



The Machine that Changed the World

- Published in 1990
- Describes lean production at Toyota and how it allowed car manufactures to create high quality automobiles
- Concepts are directly applicable to software development
- For example, don't test quality in, build it in
- Understanding these concepts will give you a better appreciation of the importance of requirements and design in software development and delivering high quality software solutions



http://www.goodreads.com/book/show/93903.Machine_That_Changed_the_World

Persistence and Transactions

- These topics are related, but not intertwined; each has a distinct role and a distinct set of issues to be dealt with
- *Persistence* is a service that enables the state associated with an object to be preserved beyond the lifetime of the object in memory
- A *Transaction* is a collection of operations that must be treated as a single unit of work
- Usually the update of the persistent state of an object is treated as a transaction, and multiple method invocations can be treated as a unit of work
- *The question is: if an object can disappear (when an application exits, for instance) then where does the persistent state live and how can we construct another object representation of the same entity from it?*

Object Persistence

- Many objects have state (i.e., data that is part of the object) that is kept in some persistent storage element such as a database or file. Such objects are said to be *persistent*
- When such an object is instantiated, the data must be fetched from the store and appropriate data elements assigned to data members of the object
- Similarly, when the state of the object has been updated appropriately in memory, the new data can be written to the data store
- The most common case involves storing the data in a relational database, leading to what is now referred to as the *object-relational mapping*
- The mapping is made more complex when more than one table is required to represent the object, when the data formats in the table(s) need to be transformed into the in-memory data types of the data members, and when multiple data bases are required
- Moreover, database connections need to be established and maintained without consuming too many resources
- A good persistence service will address many of these requirements

Transparent Persistence

- *Transparent persistence* is the storage and retrieval of persistent data with little or no work on the part of a developer
- For example, Java serialization is a form of transparent persistence because it can be used to persist Java objects directly to a file with very little effort
- Serialization's capabilities as a transparent persistence mechanism pale in comparison to those provided by frameworks that implement the JPA

What's the Purpose of the Storage Mechanism?

- Is it a minimalist role that allows one to store only that information that allows reconstruction of the original?
- Is it a "high fidelity" rendering that allows one to see the full details?
- A mapping is not acceptable if it does not allow us to reconstruct all the relationships among the objects as we read from the stored information

Things to Worry About in a Persistence Mapping

- Preserving Identity
- Referential integrity – make sure there are no dangling references or references to non-existent objects
- Associations
- Inheritance
- Mapping attributes

Other Persistence Considerations

- Object attributes that are not persistent
- Rules for updates
  - auto-updates on any state change
  - program-controlled updates
- For legacy systems, the mapping is harder
  - the mapping from classes to tables (or even to databases) is not 1-1
  - data type conversion
  - functional conversion
  - many queries and updates are performed via stored procedures
- Different roles for XML documents vs DBMS tables

Identity

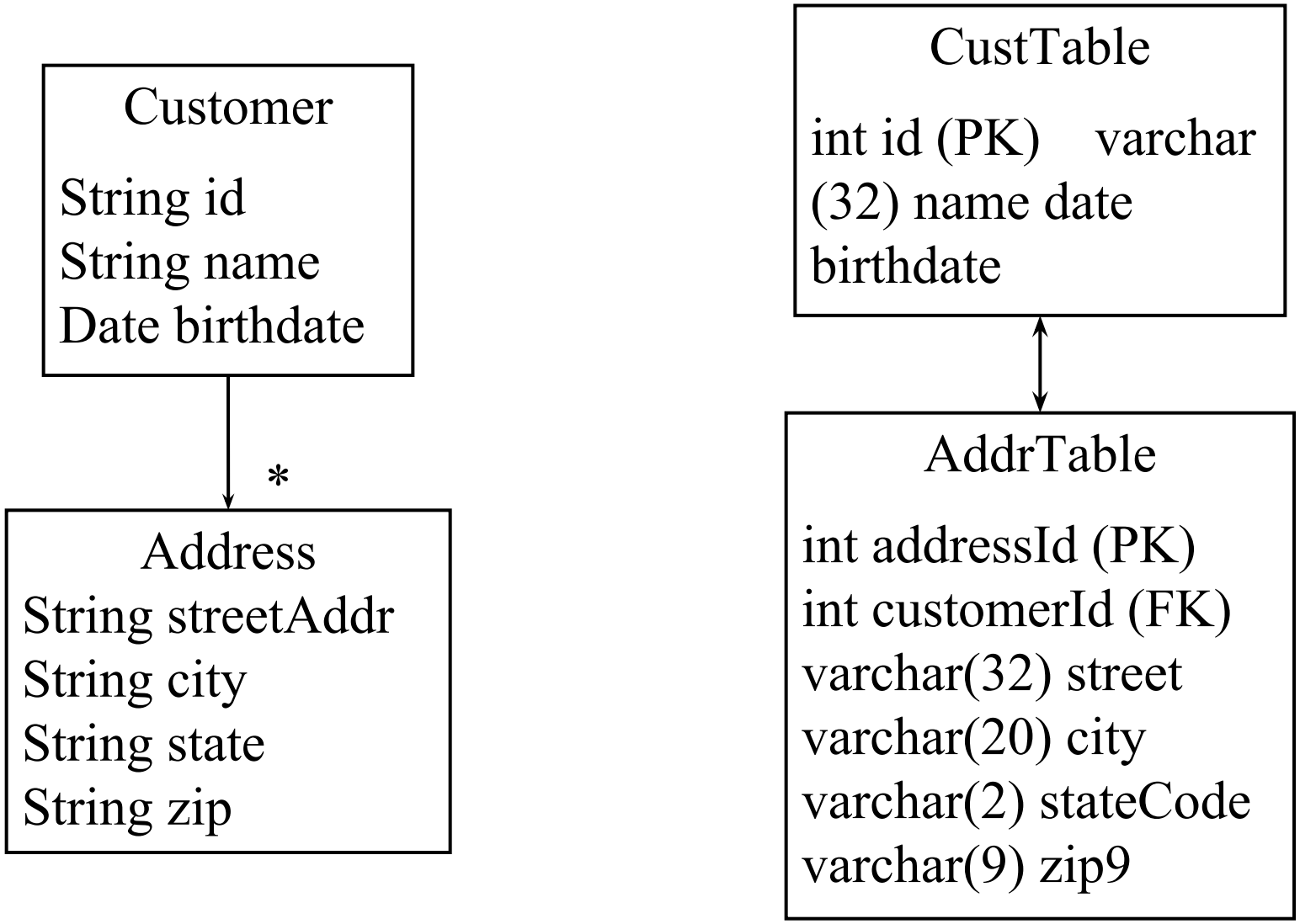
- Object identity in a programming language is pretty straightforward – one has a variable containing a reference to a particular object; the reference (in a JVM, in our case) is the identifier
- For remote objects the remote identifier is distinct from its identity in its local environment
- When we externalize objects, whether it's into XML, flat text files, or a DBMS, we have to go a different route and specify a unique identifier for that object in its context – these are typically called *keys* or *ids*.

Referential Integrity

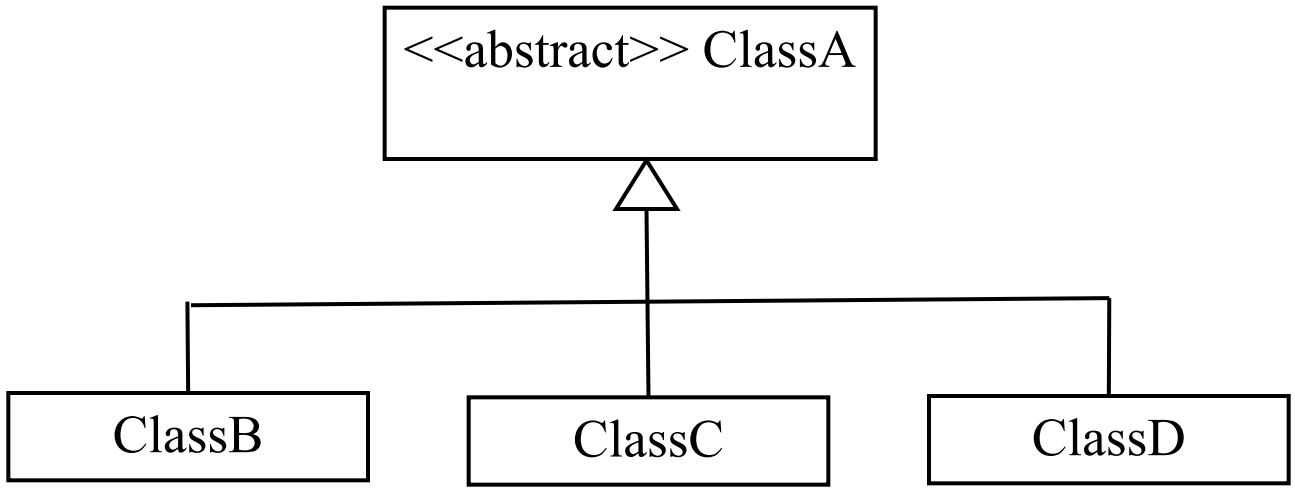
"Usually when someone mentions the term "referential integrity" only relational DBMSs come to mind. But the term is really broader and encompasses databases in general as well as programming languages and modeling. The connotations of "referential integrity" that arise from relational DBMSs are those of an implementation mechanism. However, the deeper meaning is that of dependencies among objects. In this broader sense, referential integrity is an integral aspect of how we think about problems and represent them via models. Consequently, we must deal with referential integrity regardless of the implementation platform -- RDBMS, OO-DBMS, or programming language."

- One can add XML to the list

Associations: Object Model to Relational Model



Mapping for Inheritance



- Three choices:
  - Create one table whose list of fields is the *union* of the fields in the four classes
  - Create a table for A, and separate tables for B, C, and D with foreign keys to the entries in A
  - Create a table for each concrete class, repeating the fields of ClassA

Attribute Mapping

- SQL databases have a limited set of primitive types
- There are many mapping issues to be addressed
  - Combining multiple fields into one column and splitting fields across many columns
  - Mapping data members of type enum to strings
  - For legacy databases, dealing with alternative mappings for types such as booleans

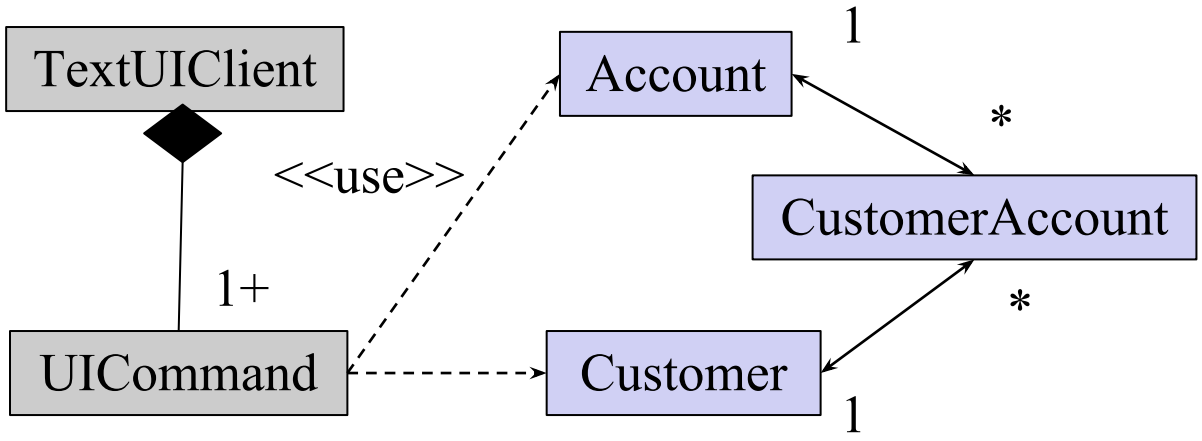
What a RDBMS Does Well

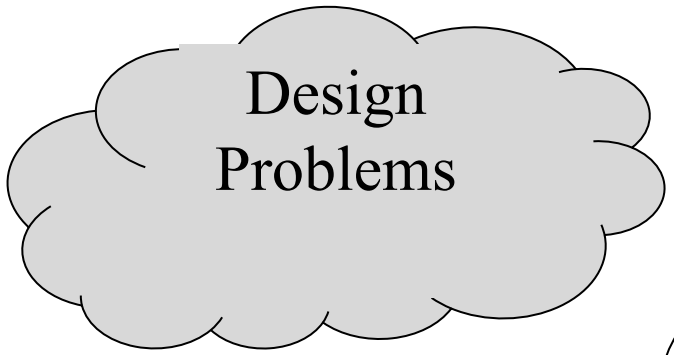
- Provides support for definition of data in the form of tables, where each field (column) can have a distinct data type, and one or more fields are designated as a primary key
- Allows definition of cross-table references (including support for referential integrity)
- Supports a very rich query capability (through SQL) for retrieving data as self-describing sets
- Supports transactional access and concurrency control, a key enabler of simultaneous access by multiple applications
- Supports other technical features such as data replication, security, backup and recovery, high performance (in terms of transactions per second), redundancy, monitoring, tuning of indices, etc
- Lots of other tools

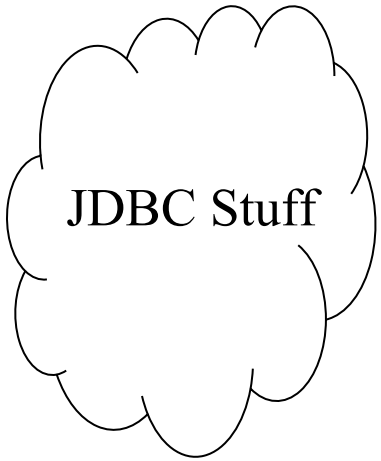
What About Persistence Tools?

- There have been many tools available for persistence for a dozen years
- Most take the point of view that you're going to create your data model from your object model or vice versa
- This automatic creation appears easy, but a good object model might not yield a good data model, and vice versa (I'd choose the former first)
- The good news is that tools that do this generation hide a lot of the nasty details mentioned before

The Gap







How could a UICommand get all Accounts related to a given Customer and use the Account objects created on return?

Some Ways to Frame the Design Questions

- Rule # 1: The domain objects don't know anything about the database and JDBC
- Rule #2: The clients don't know about the database and JDBC
- What helper object is responsible for making a request through the JDBC to get all customers and return a collection of customer objects to the requestor; what if we just want to lookup a customer by the customer id?
- What helper object is the owner of those returned objects?
- What helper object initializes the JDBC code (e.g., creates the JDBC driver) so that the data access classes can get connections and submit database requests?
- The questions are buried in the cloud on the previous slide

How Can We Handle Persistence?

- Martin Fowler has a set of patterns for rolling your own object-relational mapping in Patterns of Enterprise Application Architecture, Pearson Education, Boston, MA, 2003 (14th printing, 2008)
- One we'll look at is Table Data Gateway.
- Note: There are more powerful techniques for doing persistence.
- The 'Base Pattern' for the Table Data Gateway is the 'Gateway Pattern', which Fowler says "encapsulates access to an external system or resource" (see his catalog of Enterprise Architecture Patterns at http://martinfowler.com/eaaCatalog/)

Table Data Gateway – 1

- Problem: How to provide persistence for domain objects without mixing SQL or other database access logic with domain logic
- Table Data Gateway defines an object that acts as a go-between from domain objects to database tables. The Table Data Gateway objects provide all the access needed for that table, which can include reading lists of objects (with filtering), creating new objects, updating objects, and deleting objects [some people refer to this as CRUD operations]
- In this approach, usually there is one such object per table, though with a small problem domain, one gateway can handle multiple tables

Table Data Gateway – 2

- According to Fowler, the 'trickiest thing about a Table Data Gateway is how it returns information from a query'
- It could return the result set (in JDBC, this is the ResultSet object), it could return some kind of (hash) map object, it could return a data transfer object, or finally it could return a domain object.
- Returning a Data Transfer Object has the most advantages

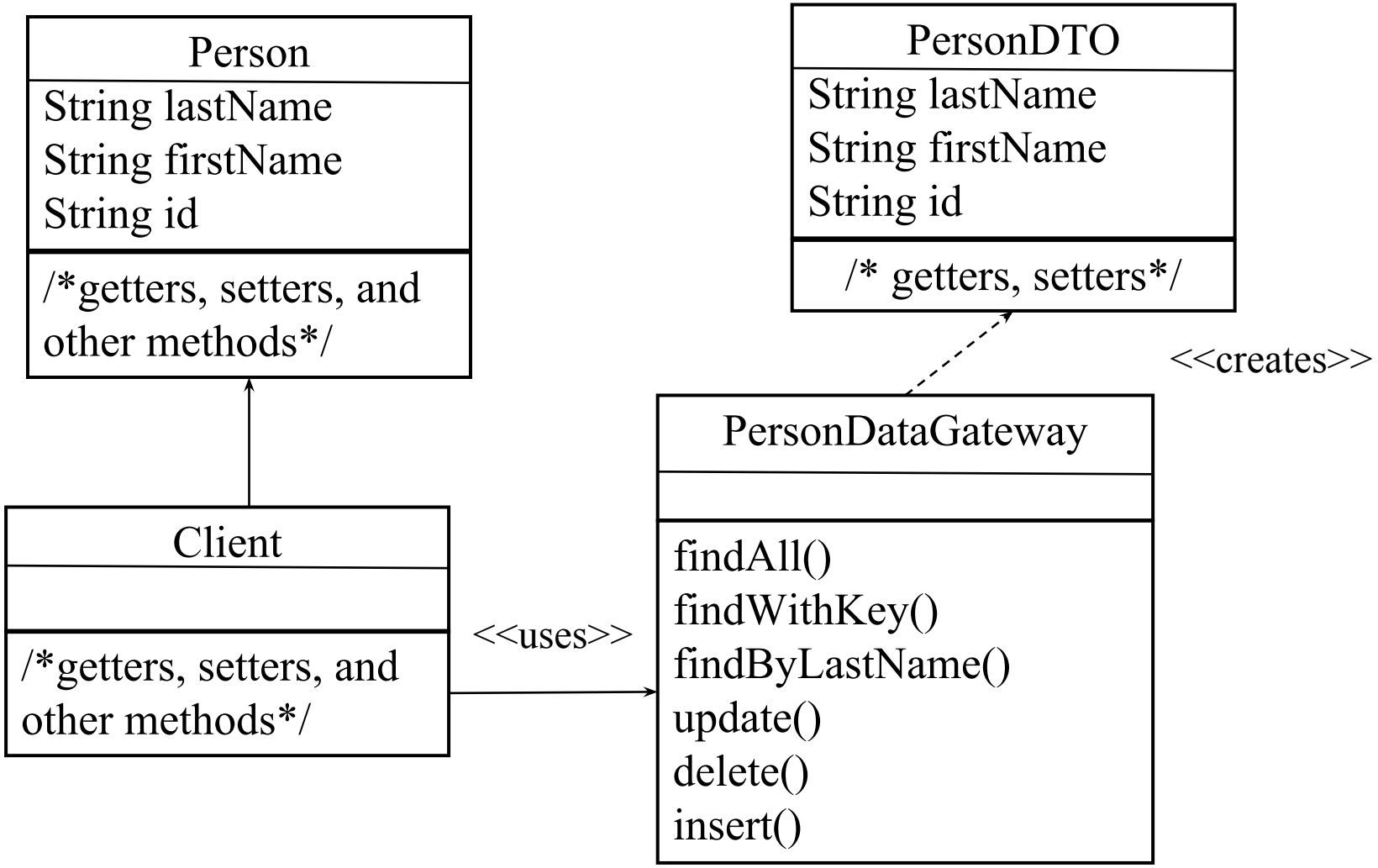
Data Transfer Objects

- Data Transfer Objects have the advantage that they match the domain object, without the behavior.
- In constructing the DataTransfer Object (DTO), one can do all the data conversion needed (e.g., from String to an enum type, from a Date object to a GregorianCalendar object, etc)
- Then 'all you need' is a factory method to create the domain object from the DTO. Which class has that factory method?

A Table Data Gateway for Person Objects

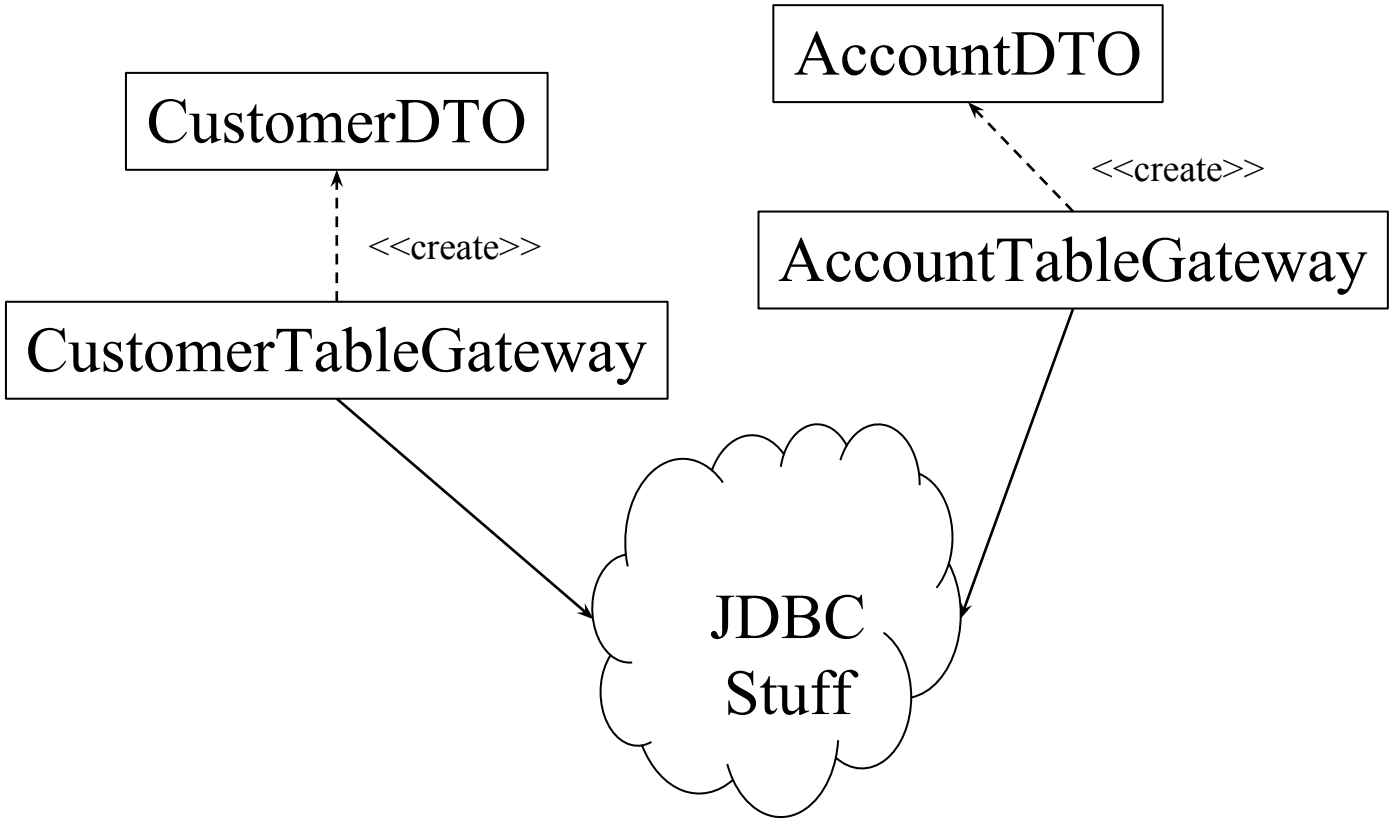
- Imagine that we have a Person domain class whose data is stored in the ‘person’ table
- We can define a PersonDataGateway that can
  - Retrieve person objects by key or other attributes, including retrieving all person objects, returning a collection of PersonDTOs
  - Create Person objects and insert them into the table
  - Update a Person object
  - Delete a Person object by key

A Simple Diagram



A Partial Data Access Service

- The Data Access Tier knows all about how to read and write data from and to the tables in the database, using the TableDataGateway Pattern. Buried beneath those classes is the JDBC bridge to MySQL



Assignment 4

Authentication Service

Review:

- Assignment
- Requirements
- Gradesheet