**Intro to Apex**

How do you delete custom objects thru coooode?

* U do not
* Can only do declaratively thru Salesforce

What are some adjectives used to describe APEX?

* Strongly-typed
  + Because it’s static-ly typed (cannot change the datatype of a var after it has been instantiated)
* Java-like
  + Syntax is similar to Java’s syntax
* Case-insensitive
* Multi-tenant aware
  + Aware that SF is hosted in a multi-tenant environment
  + Working w a database, and is aware of that
    - So no overwrite issues
    - Has inline code that allows you to commit to db so we don’t hit governor limits
      * SOQL, SOSL, and DML syntax ( is what = the inline shiz)
* Cloud-hosted

**What are the different data types that exist within Apex?**

* Primitive→ if not assigned a value, they will store null
* blob
  + Stores binary data
* boolean
  + Stores true or false
* date
  + Stores a date
* time
  + Stores a time
* datetime
  + Stores a combination of date and time
* decimal
  + Stores a non-whole number
* double
  + Stores a 64-bit decimal
* id
  + Stores a 18-character long unique id (if the 15-character is stored, it is converted to its 18-character counterpart)
* integer
  + Stores a 32-bit integer
* long
  + Stores a 64-bit integer
* string
  + Stores zero or more characters enclosed in single quotes
* object
  + Stores an instance of a class generally in our code
* Non-Primitive→ These are divisible (can be broken down into further values)
  + List
    - Essentially an array but dynamically sized
    - All contents must be of the same data type
  + Set
    - Keys
    - does not allow for duplicates
    - Unordered collection of unique values
    - All contents must be of the same data type
  + Map
    - Key-value pairs
    - Values can be duplicates, but keys cannot be
    - Keys must all be of the same data type and values must all be of the same data type but keys and
    - Unordered
  + Enum
    - Abstract Data Type with a predefined set of values that don’t adhere to a numeric order.

--------------------------------------------------------------------

* + sObject
    - Stores a custom or standard object created within Salesforce
    - *(think of it like the s stands for Salesforce--a Salesforce Object =sObject)*

**What are the control-flow statements?**

Control flow statements allow for the execution of certain lines of code

* The statements:
  + If
    - It’s a conditional logic that is applied if the condition is true.
  + if-else
  + if-else-if
  + switch
* Or repeating execution of code with:
  + while loops
  + do-while loops
  + for loops
  + list/set for loops

**What is a method?**

A method is a behavior of an instantiated object from a class.

**What is a class?**

* A blueprint to instantiate objects from
* A template for objects that contains the methods that will define an object’s behavior and the variables that will hold an object’s state

**What kind of loops exist in Apex?**

* For loops
  + Loops based on an iterating value and loops until a condition is met
* While loops
  + Loops until a condition is met, checks this condition before content is executed
* Do-While loops
  + Executes content within a loop, checks a condition, then loops until a condition is met-
* List/set(collection) for loops
  + loops through each element of a list or set. Ex: for (Integer i : acc){do logic}

**SOQL, DML, & TCL**

**Write a SOQL query to retrieve all the Opportunities in the system.**

[SELECT id FROM Opportunity]

**Write a SOQL query to retrieve all Contacts with either 2 a’s or 2 e’s in the name.**

[SELECT id FROM Contact WHERE name LIKE ‘%a%a%’ OR ‘%e%e%’]

**Write a SOQL query to retrieve all Accounts and their associated Contacts Opportunities in the system.**

[SELECT id, (SELECT id FROM Opportunities) FROM Account WHERE AccountId IN (SELECT id FROM Contact)]

**Write a SOQL query to retrieve total annual revenue of Accounts by rating, with no null values**

[SELECT SUM(AnnualRevenue), Rating FROM Account WHERE rating != null GROUP BY Rating]

**What are the DML operations?**

The data manipulation language operations available in SOQL include:

* INSERT
* UPDATE
* UPSERT
  + a combo of UPDATE and INSERT
* DELETE
* MERGE
  + combo of UPDATE and DELETE
* UNDELETE

**How can we perform a partial insertion of a collection of records?**

* Yes we can! We can use the database class methods to accomplish this. Such as database.insert() for example. We just have to remember to put the false flag on the “all or nothing” argument to make sure we allow a partial insertion.
* Database.insert(thingToBeInserted, false)
* We can also access the result like so
  + Database.SaveResult sr = Database.insert(thingToBeInserted, false);

**What 2 methods are used to perform TCL?**

We can use database.setSavepoint() and database.rollback() for transaction control language operations.

Ex: Savepoint sp = new database.setSavepoint();

database.rollback(sp);

**What are some database class methods?**

Database.insert , .update, .delete …dml methods basically.. Then also database.setSavepoint(), .rollback, .saveResult, .deleteResult, .undeleteResult, ect.

Saveresult, deleteResult, undeleteResult have methods to getID(), getErrors(), isSuccess();

**OOP Concepts, Exceptions, and Triggers**

**What are the 4 pillars of OOP?**

* **Abstraction**
  + Removing or HIDING (see, *it makes sense here -Ty*) unnecessary complexity from a user
  + Abstraction of code so that the user only sees the results, not how it does it. This is achieved in Salesforce by declaring a method/class abstract or virtual. Or declaring Interfaces.
  + Virtual classes can only contain virtual or regular methods.
  + Virtual classes can be instantiated, abstract classes cannot
  + In Salesforce, the entirety of declarative programming is abstraction at work, we are having the implementation hidden from us!
  + Uses the keyword “abstract”
    - Or “virtual”
    - Or “implements”
* **Inheritance**
  + (from pre-interview) One class (class Dog) can extend another (class Mammal). By doing so, class Dog inherits, or acquires, all of Mammal’s non-private methods and fields (behaviors and states) such as eat, sleep, age, weight.
  + Inheritance is the mechanism through which child-classes can acquire all of the parent classes non-private methods and fields.
  + “Extends” is for abstract/regular classes
  + “Implements” is for interfaces
  + Can utilize this, super keywords to access the parent or current instantiated object
* **Polymorphism**
  + The ability of an object to take many forms; Aku is an instance of the class Dog, which is a subclass of Mammal, which is a child-class of Animal. Aku is a Dog object, but she is also a Mammal object, and an Animal object. Aku is a polymorphic object.
  + Overloaded constructors can be a method of implementing polymorphism
  + Overloading & overriding helps implement polymorphism
* **Encapsulation**
  + variable scope (local variables, defined within a function—so has a pre-determined scope)
    - cannot reference a local var outside of its function
  + “Data hiding”
    - Ty h8s this
  + Access modifiers
    - Private—no other classes have access (only this class has access)
    - Protected—only the classes and its subclasses / child classes have access
      * Classes in the same package have access, too
      * Subclasses can be in any package
    - Public—all classes have access
      * Anything w/in same namespace
    - Global—any classes / code can have access (regardless of namespace)
  + Getter / setter
    - the getter and setter functions allow you to return and modify the value of a private variable stored in an object, respectively.
    - Getter and Setter are functions that modify private variables.
    - Getter / accessor—returns the value of a variable stored in an object
    - Setter / mutator—allows you to change the value
    - (from pre-interview study guide)
      * Wrapping code together into a single unit (marking a class as protected would ensure that its code could only be accessed by its child classes and other classes in the same package)
        + Classes could be in any package
        + A package is a namespace that organizes a set of related classes and interfaces

**What is unique about an Inner Class?**

* Declared within another class. A class can contain at most one layer of inner classes and an inner class cannot itself have an inner class. A top-level class can have multiple inner classes so long as they are not nested.The access modifier we provide to an inner class cannot be more permissive than the top-level class that holds it.

**Name as many exceptions as you can:**

* External services
  + CalloutException
  + EmailException
  + ExternalObjectException
* Database
  + DMLException
  + QueryException
  + SearchException
  + SObjectException
* Data types and variables
  + IllegalArgumentException
  + ListException
  + NullPointerException
  + StringException
  + TypeException
* Visualforce related
  + InvalidParameterValueException
  + NoAccessException
  + NoDataFoundException
  + SerializationException
  + VisualforceException
* Data formats
  + JSONException
  + XmlException
* Uncatchable exceptions
  + LimitException
* Miscellaneous
  + AsyncException
  + BigObjectException
  + MathException
  + NoSuchElementException
  + RequiredFeatureMissing
  + SecurityException

* Custom Exceptions - Making a new Apex class that has a name ending in Exception extends Exception superclass (*Ex. public class MyCustomException extends Exception{}* )

**What are 4 important exception methods?**

* getCause - returns cause of the exception as an exception object
* getLineNumber - returns an integer of the line number where exception was thrown
* getMessage - returns the error message that displays for the user
* getTypeName - returns the type of exception such as dmlexception, listexception ect.

**What are our trigger best practices?**

**OMBA**

* One trigger per object
* Make it Logicless triggers -- use helper classes
* Bulkify your code -- no soql /dml in loops
* Avoid recursive triggers

**What are trigger events?**

Trigger event is a combination of trigger type and a DML operation. There are 7 trigger events:

* Before insert
* Before update
* Before delete
* After insert
* After update
* After delete
* After undelete

**What are the 2 types of triggers?**

* Before
* After

**What are trigger context variables?**

* Predefined framework entities allow us to determine information about the event that invoked our trigger and work with records involved. Most important tools in the trigger toolbox. 13 context variables in total.
* Trigger.new
* Trigger.old
* Trigger.operationType
* Trigger.Size
* Trigger.oldMap
* Trigger.newMap
* Trigger.isUndelete
* Trigger.isAfter
* Trigger.isBefore
* Trigger.isDelete
* Trigger.isUpdate
* Trigger.isInsert
* Trigger.isExecuting

**What is the annotation used for declaring a method or class as part of a test?**

* @isTest

**What is the annotation used for making a private method visible?**

* @testVisible --used in class being tested; allows test class to utilize those methods/variables

**What are ways to create test data for test methods?**

* You can use a test setup method.
  + Ex: @testSetup static void setup() {generate and insert data here to be used by other methods in the class}
* You can also load a CSV of data.
  + Ex: List<sObject> ls = Test.loadData(Account.sObjectType, 'testAccounts');
* You can generate the data within the method itself.

**What is the parameter that controls access to your org for a test?**

* @isTest(seeAllData=true)

**What are the code coverage percentages?**

* Code coverage percentage is the percent of lines of code that have been covered by the test
* 75% for classes
  + Need 75% to meet salesforce’s requirement in order for the code to be but into production
  + Also applies to trigger handlers, since we make trigger handlers as their own class
* and 1% for Triggers
  + ????

**What are the testing best practices?**

* Very minimum is to have 1 positive or one negative test case and 1 single or bulk test case, good is 4 excluding restricted user, best is all 5
* Positive (start w this one)
  + ensure that the expected behavior occurs whenever the user correctly makes use of it
  + doing what it’s supposed to do
  + what you expected actually happens
  + if everything is correct, we get the correct answer
* Negative
  + ensure that if things are not correctly used, we are handling it gracefully
  + the errors being returned are the correct errors
* Single action
  + Does your code work for a single record?
* Bulk testing
  + Does it work when we put in 200 records? scalability , governor limits
* Restricted user
  + Are we sharing or not sharing access in an appropriate way
  + we wanna have all 5, but could technically reach 100% with just one
  + HAVE ALL 5; we ain’t lazy

**What are some test execution environments that we can use for testing?**

Dev console, vscode, test execution ui, and running test api