**Challenge 2 Answers - Norm Gershon**

**0.1 What to expect from this course 3m 6s**

What is the intended purpose and potential advantage of learning object oriented design?

To provide a structure for better organization and clarity in your code, as well as improved communication of concepts.

**0.2 Exploring object-oriented analysis, design, and development 1m 41s**

Why might it be advantageous to analyze and design before beginning programming?

Better organized, focused, and efficient approach.

**0.3 Reviewing software development methodologies 4m 8s**

What is the difference between a "waterfall" and an "agile" approach to development? What

is an iteration and how do we to use them to build software?

Waterfall is a rigid, step-by-step approach, not responsive to changes. Agile is a flexible, iterative approach that allows ongoing design changes to be incorporated into the process. Each iteration doesn’t have to be perfect, allowing you to review where you’re at and where you’re going, and make changes, before proceeding to next iteration.

**Section 1 - Core Concepts**

**1.1 Why we use object-orientation 2m 42s**

What are the various types of programming languages and in which domain is each used?

Procedural, functional, logic languages may be used in more academic or specialized environments. Object-oriented languages are more practical for, web, mobile, game development where objects, with their self-contained data and logic, better represent how you’d think about problem you’re trying to solve.

**1.2 What is an object? 5m 22s**

Describe in your own words the three properties of a computing object.

Something that has its own identity that can be described by various characteristics and behaviors.

**1.3 What is a class? 4m 43s**

Explain how classes are analogous to blueprints. Include the relationship between a class and an object. Can you think of how the analogy breaks down?

A blueprint is a template for creating something with specific attributes and behaviors. Analogy weakens in sense that class could allow greater variations than could be represented with a blueprint.

**1.4 What is abstraction? 2m 45s**

When a developer uses the term "abstraction" what are they describing?

Focusing on the core essential qualities pertinent for that particular project, avoiding irrelevant info.

**1.5 What is encapsulation? 3m 45s**

What does encapsulation prevent? What does it enable?

It prevents changes in one part of the application from cascading down and requiring alterations in other parts. Enables you to more safely change the way an object works; useful when creating other classes.

**1.6 What is inheritance? 3m 35s**

Describe the inheritance relationship between classes. When would this relationship be

advantageous to establish?

Child class has all attributes/methods of Parent class. Advantageous when Subclass needs all attributes/methods (and changes to these) of Superclass.

**1.7 What is polymorphism? 3m 22s**

What is the basic idea behind polymorphism? How can it make the classes we create more

flexible?

Lets the program perform the correct behavior when what you’re giving it could take different forms. Allows specialization in different classes so that you can do something with an object from any class and the correct thing will happen.

**Section 2 - Object-Oriented Analysis and Design**

**2.1 Understanding the object-oriented analysis and design processes 4m 13s**

What are the steps of analysis that come before writing code for an application? Why do you

think these steps make writing the code easier?

1. Gather requirements
2. Describe the app
3. Identify the main objects
4. Describe the interactions
5. Create a class diagram

These steps help you get organized and focused.

**2.2 Defining requirements 6m 9s**

What should you have after you've completed the first phase of defining your requirements?

At least a list of the minimum requirements for your application.

**2.3 Introduction to the Unified Modeling Language (UML) 1m 54s**

What is UML? Why Is it useful to visualize your application before coding it?

Common diagramming techniques for object-oriented system. Useful for organization and understanding of your design.

**Section 3 - Utilizing Use Cases**

**3.1 Understanding use cases 6m 11s**

Write a use case for creating an event on your phone's calendar.

Title: Create calendar event

Actor: User

Scenario:

1. User selects calendar icon
2. User selects date
3. System presents date activity and add/delete/edit options
4. User selects “add event”
5. System presents template
6. User completes form
7. System confirms event added and presents updated date activity

**3.2 Identifying the actors 4m 16s**

Can you think of a use case for a mobile application in which the actor is not the user of the

mobile device?

Field inventory app gets polled by headquarters.

**3.3 Identifying the scenarios 5m 7s**

Write another use case for a mobile device user interacting with a calendar application. This

time include a couple extensions when crafting your scenario.

Title: Create calendar event

Actor: User

Scenario:

1. User selects calendar icon
2. User selects date
3. System presents date activity and add/delete/edit options
4. User selects “add event”
5. System presents template
6. User completes form
7. System confirms event added and presents updated date activity

Extensions: Another event scheduled at same time

1. User accepts simultaneous event
2. User chooses different time

Extensions: Calendar has not synced with main calendar

1. Sync now
2. Cancel event add
3. Proceed anyway

**3.4 Diagramming use cases 4m 18s**

Do a google image search for "use case diagram." Notice how many variations there are.

What do they all generally have in common?

Lines connecting actors to use cases; box around system.

**3.5 Employing user stories 3m 43s**

Write 5 user stories to describe a mobile user interacting with his or her maps application.

1. As a user, I want to be able to get directions so that I know how to get from one place to the other.
2. As a user, I want the app to know where I am on the map to save me time and effort.
3. As a user, I want to know where all the coffee shops are so I can get my fix.
4. As a user, I want to see the route of my journey so I can get an overview of my trip.
5. As a user, I want to be able to turn on city borders so I know where towns are located.

**Section 4 - Domain Modeling (Modeling the App)**

**4.1 Creating a conceptual model 1m 59s**

Just let it soak in. No questions here.

**4.2 Identifying the classes 2m 27s**

Identify the classes in the use case you constructed for a user interacting with his or her

calendar application in chapter 3.

User, calendar, date, event, confirmation

**4.3 Identifying class relationships 2m 38s**

Identify the relationships among the classes you found above. Create a conceptual model

where you diagram these relationships and then upload a picture of your model below.

Notification

Date

Event

Calendar

User

uses contains

adds contains

**4.4 Identifying class responsibilities 6m 43s**

Identify the responsibilities of the classes you found above. List them here.

Calendar: display dates

Date: display date event details, present options to add event

Event: set event details, confirm event, denote schedule conflict

Notification: display confirmation, display conflict notification

**4.5 Using CRC cards 2m 49s**

If you'd like, try creating CRC cards for the model you made above. There's no need to

respond here, just try it out and see if you like this form of organization.

**Section 5 - Creating Classes**

**5.1 Creating class diagrams 6m 11s**

Construct Class Diagrams for the classes you imagine exist in a twitter app, a maps app, a

calendar app, or any other app you would like to make. Do you find that it is easier to come

up with the attributes or with the behaviors? Why do you think that is?

Easier to list behaviors, to think of where you want to go as opposed to how to get there

|  |  |
| --- | --- |
| **Location** | **Routes** |
| coordinates: float | locations |
| displayMarker() | createGrid()  displayGrid()  calcRoutes()  displayInfo() |

**5.2 Converting class diagrams to code 4m 57s**

How might the separation of interface and implementation in Objective-C be an advantage

when working with class diagrams?

They follow a similar separated format.

**5.3 Exploring object lifetime 5m 55s**

Recall from Stage 1: What are the constructors and destructors in Objective-C? Why do we

use them?

alloc, init, free. Constructor used to instantiate an object; destructor used mostly to free resources.

**5.4 Using static or shared members 5m 22s**

Like the interest rate example in the video, give three additional examples of data that would be the same for all instances of a class.

Class UPSDriver : uniformColor

Class Vampires: preferredFood

Class Prisoners: minExerciseTimePerDay

**Section 6 - Inheritance and Composition**

**6.1 Identifying inheritance situations 6m 49s**

Describe in your own words what inheritance is and how it is useful when constructing

classes.

A property which allows you to utilize attributes and behaviors which are pre-defined in the superclass(es), so you don’t have to reinvent the wheel every time you define a class.

**6.2 Using inheritance 2m 43s**

Referring to the apps on your phone, come up with three examples where you believe

methods are being inherited from superclasses and called by subclasses.

1. Phone rings call to Sounds method
2. Map app calls to Display method
3. Calculator calls to Math method.