Challenge 2: Time To Dig In

Section 0 - Introduction

0.1 What to expect from this course?

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

Answer: “ *The main purpose is to build better apps. To make them faster and more flexible. ”*

0.2 Exploring object-oriented analysis, design, and development?

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

*A:” First of all it gives you an understanding of the problem, give you an idea what to do and how to design a solution by creating diagrams, sketches on the whiteboard, written descriptions. The more you understand the analyze and design, the easier the programming will be.* ”

**0.3 Reviewing software development methodologies?**

What is the difference between a "waterfall" and an "agile" approach to development? What is an iteration and how do we use them to build software?

A:” ~ *The main advantage of the Agile model is the backward scalability. Under Waterfall approach it is not easy to change decisions which were made in the earlier stages. Any changes in the waterfall model demands the building of the entire project right from the start.*

*~ The flexibility of error checking during any phase of development makes the Agile model the best approach for less erroneous results as compared to Waterfall model. The Waterfall model can only test bugs after the development of the modules.*

*~ Agile model provides flexibility for making changes as per customer needs and guarantees client satisfaction. The Waterfall model suffers drastically in this respect since it does not allow any modifications midway through the development process.*

*~ Any development will involve several incremental cycles, iterations, each including analysis, design, and programming*.

Section 1 - Core Concepts

1.1 Why we use object-orientation?

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

A: “*Fundamentally, languages can be broken down into two types:* ***imperative*** *languages in which you instruct the computer how to do a task, and* ***declarative*** *languages in which you tell the computer what to do. Declarative languages can further be broken down into* ***functional*** *languages, in which a program is constructed by composing functions, and* ***logic*** *programming languages, in which a program is constructed through a set of logical connections. Imperative languages read more like a list of steps for solving a problem, kind of like a recipe. Imperative languages include C, C++, and Java; functional languages include Haskell; logic programming languages include Prolog.*”

1.2 What is an object?

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

A: *” a) identity. Each object has its own identity. And they are not just physical items which we can see or hold in real life, that can be anything else which has attributes and behavior;*

*b) attributes- information that describes a current state, like describing a color or giving other sort of clue;*

*c) behavior- things that attributes can do. ”*

*1.****3 What is a class?***

**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*:” Classes are symbolic representations of objects; classes describe the properties, fields, methods, and events that form objects in the same way that blueprints describe the items that form buildings. Just as a blueprint can be used to create multiple buildings, a single class can be used to create as many objects as necessary. Just as a blueprint defines which parts of a building are accessible to people who use the building, so too can classes control user access to object items through encapsulation. ”*

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

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

A: *” They describing the essential qualities of something, of an idea rather than one specific example, automatically discard what’s irrelevant. And how the class should look like under specific circumstances at that time by focusing just on the essentials. ”*

**1.5 What is encapsulation?**

What does encapsulation prevent?

**A**: *” Hiding the internals of the object protects its integrity by preventing users from setting the internal data of the component into an invalid or inconsistent state. A benefit of encapsulation is that it can reduce system complexity, and thus increases robustness, by allowing the developer to limit the inter-dependencies between software components.*

**1.6 What is inheritance?**

Describe the inheritance relationship between classes. When would this relationship be advantageous to establish?

A*:” Inheritance is a great form of reuse. Instead of creating classes from the scratch which have similar attributes and behaviors, we can create a superclass or parent class and subclasses or child classes which are going to inherit each characteristics from main one.*

*This would be really advantageous by saving our time…*

**1.7 What is polymorphism?**

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

A: *” Polymorphism is the ability to create a variable, a function or an object that has more than one form. It lets us automatically do the correct behavior even if what we are working with could take one of many different forms. Polymorphism lets us work freely with any objects that have been created from any of the classes, which makes them more flexible. “*

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?**

A*:” There are 5 steps before writing code for app:*

*a) Gather requirements.*

*What the app need to do?*

*What problem are you trying to solve?*

*It is important to be specific here and write everything down.*

*b) Describe the app.*

*Have to be as easy as you can, build conversational language explaining how people would use an app.*

*c) Identify the main object.*

*This is a starting point to identify the actual classes.*

*d) Describe the Interactions.*

*This step gives us a better understanding of the responsibilities of the different objects, the behavior they need to have and what they do when interact.*

*e) Create a class diagram.*

*That would be a visual representation of the classes you need.*

*What else do you need to make your life easier after completing all those steps.*

**2.2 Defining requirements**

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

A*: “ The core of my requirement is Fundamental requirements: answer what are the features and capabilities of the application. What is the application required to do, what must it do.*

*Additional requirements would be Non- functional requirements:*

*What help do you need : about documentation, legal, performance, support or even security.*

2.3 Introduction to the Unified Modeling Language (UML)

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

A: *“ UML is nor a programming language it is a graphical notation specifically for drawing diagrams of object- oriented systems. UML describes over a dozen different diagrams. It is very useful to visualize your application because it will give an idea that this is readable and understandable whether programming language you use.’’*

Section 3 - Utilizing Use Cases

**3.1 Understanding use cases**

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

A: “

*Title:* creating an event

*Actor:* calendar user

*Scenario:* ~ user chooses a date

~ user chooses a time

~ user chooses a type of event

~ user creates an event by entering a data

~ system confirms it done

~ system will send a notification by email

**3.2 Identifying the actors**

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

*I have “ USA TODAY” and “ BBC NEWS” apps in my phone, so the editor group, people who write the news and add them are not going to be a user of the mobile device…*

**3.3 Identifying the scenarios**

**Write another use case for a mobile device user interacting with a calendar application. This time include a couple extensions when crafting your scenario.**

*A:” Title:* creating an event

*Actor:* calendar user

*Scenario:* ~ user chooses a date

~ user chooses a time

~ user chooses a type of event

~ user creates an event by entering a data

~ system confirms it done

~ system will send a notification by email

*Extension:* user choses to create an event in past time

~ system gives a push notification if that’s correct

*Extension:* user doesn’t have an email address

*~* system move forward and finishes a task”

**3.4 Diagramming use cases**

**Do a google image search for "use case diagram." Notice how many variations there are. What do they all generally have in common?**

A: *“ Yes, I’ve mentioned how many of them and all them have similarities: one or multiple actors (represented as stick figures), and use cases (represent as ellipses bounded in box).*

**3.5 Employing user stories**

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

A: *“ ~As a user*

*I want to bookmark most commonly used addresses*

*So that I don’t need to type them again;*

*~ As a user*

*I want automatically define my location*

*So that I don’t need to find this out;*

*~ As a user*

*I want different options of how to how to get a destination*

*So that I can choose what I like;*

*~ As a user*

*I want to know the traffic information*

*So I can avoid problem areas;*

*~ As a user*

*I want to know whether my destination is on a left or right*

*So that I can precede and take a right lane.*

Section 4 - Domain Modeling (Modeling the App)

4.1 Creating a conceptual model

Just let it soak in. No questions here.

4.2 Identifying the classes

Identify the classes in the use case you constructed for a user interacting with his or her calendar application in chapter 3.

date

time

event

type

system

email

user

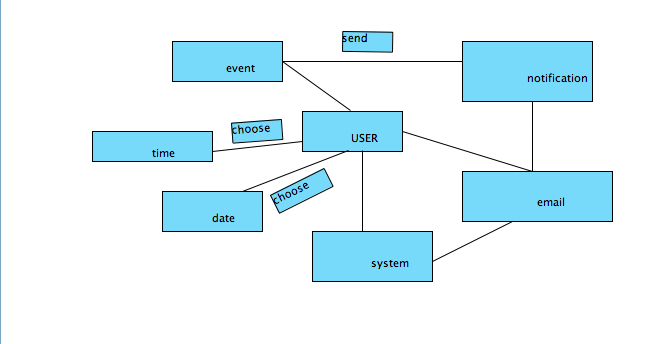
v

v

notification

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.



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

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

A: *” User choose date and time;*

*User creates type event;*

*Event sends a notification to email box.*

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**

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?

A: “ I took as an example a Facebook class Photo

|  |
| --- |
| Photo |
| nameAlbum: string  launchDate: date |
| setAlbum()  getAlbum()  getAlbumByDate()  previousPhoto()  nextPhoto()  likePhoto() |

*By experiencing contracture of class diagram I can tell that to write the attributes was much easier, because you have an idea how it looks like and what this consists of, but as long as you go to write a behavior it took longer.*

**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?**

*A: ” Separating interface from implementation means that method you provide in your object should not expose the internal implementation details of the object. If you change the internal implementation of the object, you end up having to change the interface as well. Once you change interface, you have to change all the code that uses the interface. This type of change could then ripple through the entire project. Separating interface from implementation is desirable for achieving flexible, extensible, portable and modular software. If client code depends only on the interface to an object and not on the object’s implementation, a different implementation can be substituted and the client code continues to work, without change or recompilation. Furthermore, the client code continues to work on objects supporting an expanded interface.”*

*Interface is what you can see, like the interface of windows has lots of icons you can click on and a desktop...*

*Implementation is making your program. To implement something is to make it. So like if it says: "how will you implement your program" it means "how will you make it so it works"*

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

**What are the constructors and destructors in Objective-C? Why do we use them?**

A: *“ Constructor is a special method that exists to construct the objects. The constructor will be called when the object is created. By using this we make sure that variables belonging to specific object are going to be immediately set to the right values as soon as that object is created. The destructor is going to be called when an object is being deleted/ deallocated/ released. It used when you have an object that is holding a resource, you want to make sure that that object has released any connection that has before its destroyed.*

**5.4 Using static or shared members**

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.

A: *“ 1. “Tax” is shared variable for “Food” class;*

*2. “ Retire age” is static variable for “ Women ” class;*

*3. “ Minimum wage” is static variable for “ Income” class.*

Section 6 - Inheritance and Composition

6.**1 Identifying inheritance situations**

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

A*:” Inheritance describes “****IS A****” relationship. If question makes sense for you, that might be a inheritance, something that things can share- attribute and behavior. If few classes have something similar between, we can combine that similarity and define a superclass, but at the same time be able to change and work with any of particular child classes. Its very handy, but this should occur naturally.*

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.

A: “ *Skype application :*

Superclass- “contacts” ,subclasses “online”, “recently contacted”, “skype contacts” .

*Mailbox application:*

*Superclass- “Mailboxs”, subclasses “inbox”, “vip”, “sent mail”..*

*Bank of America application:*

*Superclass- “My account”, subclasses “saving”, “credit”…*