Week 11

ASSIGNMENTS
h8 available soon and due before 10pm on 4/8 (w11)
h10 available _____ and due before 10pm on 4/22 (w13)
p4 due before 10pm on Tuesday 4/16 (w12)
p5 due before 10pm on **Friday 4/12** (w11 - install JavaFX and complete in week 11)
p6 due before 10pm on **Friday 5/3** (w14 - start in-class in and complete in week 14)
Team Project: QuizGenerator
- **milestone 1 design**: due before 10 pm on Thursday 4/18 (w12 - our design available on 4/19)
- **milestone 1 GUI**: due before 10pm on on Thursday 4/25 (w13)
- **milestone 3 final program**: due before 10pm on Thursday 4/2 (w14)

Peer Mentors: JavaFX, JSON parsing (p4 and team project), exam review.
Zhiyue (W and F 10-12 in 1289) and Yiye (Th 12:30-2:30pm in 1358, F 9-11 in 1358)

Module: Week 10 (start on week 11 before next week)

Read Team Project and install JavaFX before lecture Week 11

THIS WEEK
- Install Java FX (before lecture if possible)
- Graphic User Interfaces (GUIs)
- JavaFX Demo
- MyFirstJavaFX (p5 in class)
- Intro to Team Project
- Object-Oriented Design
- UML Diagrams
- Makefile
- Shells
- vim

NEXT WEEK
- more Java
  - enumerations
  - interfaces
  - lambda expressions

Install e(fx)clipse

Eclipse Eclipse IDE for Java Developers
Version: Oxygen.2 Release (4.7.2)
Build id: 20171218-0600

If your version of Eclipse is different, these steps will likely still help.

CAUTION: These instructions assume you are sitting at the terminal.
They will not work from a remote terminal. That means that you can not do this from a ssh connection unless you have downloaded, installed, launched, and configured an X-server and your remote connection correctly.
We recommend that you complete this project from a CSL computer.

1. Launch Eclipse and select existing or create a new workspace
2. Install the Java FX project wizard
    a. Go to Help -> **Install New Software**
    b. Work with: **--All Available Sites--**
    c. In the Details section of form
        i. Unselect **Group Items by category**
        ii. Select **Show only software applicable to target environment**
    d. Type **e(fx)clipse** in the search field
    e. Wait for listings to populate
    f. Select **e(fx)clipse - IDE**
    g. Click **Next** button to start installation
    h. Wait (patiently) for installation to complete (it can take a few to several minutes)
    i. Review Install Details
    j. Click **Next**
    k. Review Licenses
    l. Click **I accept terms of the license agreements**
    m. Click **Finish**
    n. Wait patiently (or impatiently)
    o. Click **trust certificates**
    p. Click **Accept Selected**
    q. Click **Restart Now**

Create your first JavaFX project

1. Create a Java FX Project
    a. Select File -> New -> Other
    b. Select **Java FX** -> **Java FX Project**
    c. Set Project Layout: **Use project folder as root for source and class files**
    d. Select **Finish** to create the project
2. Run your Java FX application
3. Close the GUI program window
4. Proceed to learn about Java FX controls.

Graphic User Interfaces (GUIs)

Idea:

- Create a user interface that user can interact with
- click bottom, cover, right-click, double click, click and drag
- Macintosh, Windows 2.2, Linux - Ubuntu
- Pro: see your option, no command memorizing.
- Con: hard to test, slow, know where option, not all options make it to GUI.

Java FX — Java latest GUI.

AWT - abstract windowing toolkit (classes that create graphic UI components)
Applets - run java code in a web browser — sandbox
Swing - encapsulated common gui components - easier to use than awt packages
Java FX - latest version, has support for desktop computers and web browsers

RIA: Rich Internet Application.

Why Java FX?

- leverage familiar with Java
- rich set of UI controls.

Other Tools

- FXML.
- CSS-like html style format
- SceneBuilder

Java FX Basics

- Package application · leave this declaration
- class main extends Application — contains main method
  - can use command line args to pre-load data structure
  - then, launch(args)
- stages - Primary stage - contains a scene
- Scene - contain layout manager
- Layout managers: contain UI control

MyFirstJavaFX

In-class demonstration of Java FX project and build process

What are the properties of and use of these layout managers?

BorderLayout - default, top bottom, left, center, right.
HBox - horizontal box - displays in list order across the row
VBox - vertical box - displays in list order down "col"
GridPane - place controls at desired x, y coordinates. GUI: (0,0) top left.
Flow layout -

Create and use the following user interface controls

```
stage.setTitle

Label
Button
TextField
TextArea
ImageView
ComboBox
```

import: javafx.scene.control

ImageView ← Image ← data file

```
Image image = new Image("photo.jpg");
ImageView imageView = new ImageView(image);
```

Team Project Overview

Design and implement a quiz program. See assignment for full details.

Team Project: Requirements

- must join an A-team on Canvas (ateam N -- N is team number 1-200)
  - team must have 4-5 students
  - students may be from any current lecture
  - JOIN YOUR TEAM TODAY !!!
- **must be a JavaFX program with an interactive graphical user interface (GUI)**
- **must read in data from a json file in the provided format**
- **must write data out to a json file in the provided format**
- must present user with a drop down list or search field for question topics
- must allow user to select a number of questions, N, for their practice quiz
- must present the user N questions in some random order
- each question must have a correct answer and may have an image
- if the question has an image, the image must be displayed to the user
- the user must be able to select an answer choice or type the correct answer
- the program must track the results
- after all questions have been asked and answered,
  the results must be shown to the user both as number of questions correct
  and a percentage correct.

There are three milestones that must be completed and submitted by each Team.

Milestone #1 Design (x4)

Complete a design document that shows how your team intends to solve this problem. We fully expect your design to change as you implement it. But, you must start somewhere and documenting your expected classes, test cases, and UI components is a good place to start.

Milestone #2 GUI - looks good, doesn't work

Submit a JavaFX program that works well enough to pop up a dashboard interface, or the start of a quiz. Data may be hard coded to produce this part, and the user interface (UI) does not have to be fully functional at this stage.

Milestone #3 Working Program working program

Complete and submit your final JavaFX program that presents the user with a GUI to configure, generate, and complete a list of quiz questions. It must allow the user to select one or more topics from a list of question topics, and a number of questions to be quizzed on for practice.

Object-Oriented Design: Classes and Objects

Finding the Classes (and Objects) – Fish Tank simulation

TOP DOWN Approach
↳ Break the problem down to smaller pieces

Bottom Top.
↳ create small pieces
↳ combine into to larger / more functional pieces

1. read the problem discription.
problem domain
2. Identify (name and describe real world objects)
people, places, things
roles -
input, output data, result
3. refine the list
a. remove duplicate.
b. determine if nouns are objects (instance), class type
c. for each type – prim or reference type, class, interface, enum, abrast class.

Class Summary

type	name	description of purpose or use
class	Fish.	represent a single fish. in simulation color, location, (x, y, z), size. direction, (image) change name, points (energy)
class	water	background color, change types ... .
"	plant	subset of decoration
"	decoration	
	shark	
	Turtles	

abstract class ⇒ TankObject location, direction, color, image.
Interface movable move( )

Diagrams

hide details - focus on bigger questions.

- Document - our design choices
- communicate - our choices, ourselves, boss, team, customers (stakeholders)
- can be any format that team understand (project lead, programmers, customers)

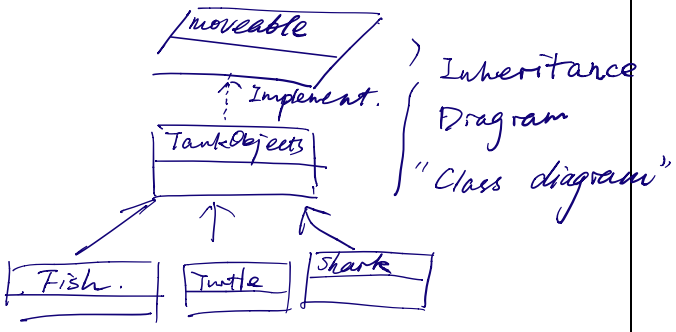
Unified Modeling Language https://www.uml-diagrams.org/uml-25-diagrams.html

Class Diagrams

shows fields, constructors, methods

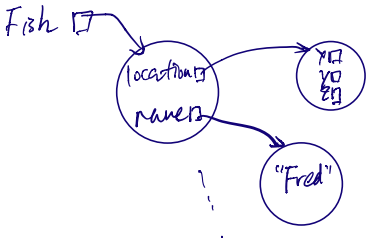
name : type

class Fish
- location : Location - direction : Direction - color : Color (RGB) - name : String } fields
+ Fish (String name) : Fish } constructor
+ move (Direction) : void + eat (Food) : int + getPoint() : int + getName : String } methods

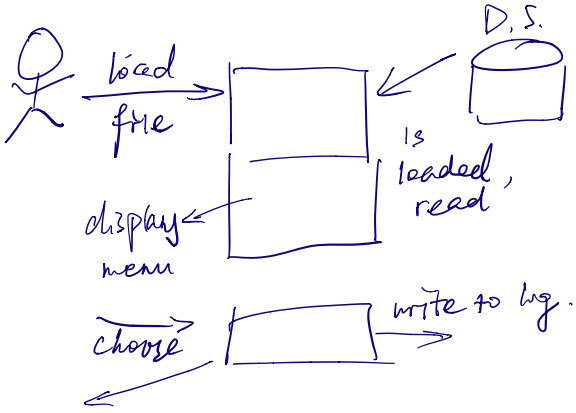


Object Diagram

Show current state of memory for an object.



Use Case Diagram



Enumerations

https://docs.oracle.com/javase/tutorial/java/javaOO/enum.html

What are they?

named constants created for you by enum

- user provides lists of names
- enum assigns integer constants values to each name
- user - use the named constraints.

Example

0 1 2 3 4

```
enum WeekDay { MON, TUE, WED, THU, FRI }
```

↑ list of name

final int MON = 0;

Enumeration Example

```
class Fish implements Moveable {
    static final int UP      = 0;
    static final int RIGHT   = 1;
    static final int DOWN    = 2;
    static final int LEFT    = 3;
    static final int CLOSER  = 4;
    static final int FARTHER = 5;
    int x = 0, y = 0, z = 0;
    void move() {
        direction = (int)(Math.random()*6);
        switch(direction) {
           case UP:      y--; break;
           case RIGHT:   x++; break;
           case DOWN:    y++; break;
           case LEFT:    x--; break;
           case CLOSER:  z--; break;
           case FARTHER: z++; break;
        }
    }
   …
```

enum directions {UP, Right, DOWN, LEFT, Closer, Farther};

Direction.UP

Rewrite using an enum for the constants of the Fish class

Makefile

```
.PHONY = run exe clean make jar runjar test clean

CLASSPATH = ./:./classes:./json-simple-1.1.1.jar\
./junit-platform-console-standalone-1.3.2.jar

make: *.java
     javac -cp $(CLASSPATH) -d . application/*.java
     java application.Main

jar:
     jar -cvmf manifest.txt executable.jar .

runjar:
     java -jar executable.jar

test: *.java
     javac -cp $(CLASSPATH) *.java
     java -jar junit-platform-console-standalone-1.3.2.jar \
 --class-path $(CLASSPATH) -p ""

clean:
     \rm *.class
```

First target; rule

← current directory

all tests current directory

```
VAR_NAME = replacement text for that variable

target : dependency
<Tab>rule action 1
<Tab>rule action 2
```

← command line "build things"

- target - name of a list of rules
- dependency - file(s) or other targets that a rule depends upon
- rule - actions to execute if the target is invoked

Shells

A shell is a command-line interpreter that displays a "ready" prompt and executes the shell command or script that is entered.

sh - Bourne shell commands
csh - C-style shell commands
tcsh - C-style shell commands
bash - Bourne Again SHell

Shell Scripting

example /p/course/CS400-deppler/public/html-s/code/Shellscripting_Examples.

Write scripts (files with shell commands in them) to

- build and run a programs
- set file and directory permissions
- create project directory structures
- read and use configuration files
- copy files
- move files
- read files
- write files

- Install GitBash (Windows)
- Enable Linux (for windows)

Examples

List the user and group permissions for the current directory

```
fs la .
```

Create a permissions group for your team

```
pts creategroup deppeler:myteam
pts adduser kuemmel deppeler:myteam
```

Create project directories for your programs — tcsh commands

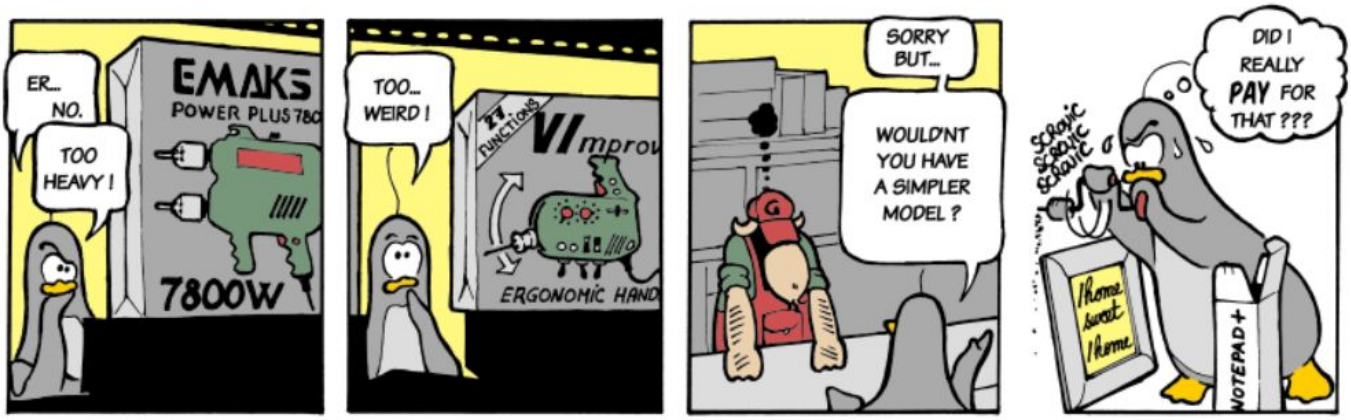
```
foreach p ( p1 p2 p3 p4 p5 p6 )
foreach? mkdir $p
fs sa $p deppeler:myteam write
fs sa $p system:anyuser none
end
```

vim

What Vim Can Do

Vim is an advanced text editor that seeks to provide the power of the de-facto Unix editor 'Vi', with a more complete feature set. It's useful whether you're already using vi or using a different editor. Users of Vim 5 and 6 should consider upgrading to Vim 7. The main advantages of Vim 6 compared to Vim 5 can be found on this page.

A General Overview



I use vim to edit pages, scripts, data files

Here is just enough VIM to get started and stuck!

command-mode - use arrow keys to move around doc

ZZ	- save current file and exit vim
u	- undo last action
i	- change to insert mode
114G	- go to line 114
x	- delete current character
cw newword	- replace current word with "newword"
dw	- delete current word
dd	- delete current line
yy	- yank current line (like cut)
p	- put "yanked" characters, words, or lines after current position
P	- Put "yanked" characters, words, or lines before current position

xp - current character, paste after

- cut & copy

- paste

insert mode

Esc	- escape insert mode and return command mode

type the text contents of your file - escape insert mode to return to command mode

to replace all ^M in a text file, the following while in command-mode

```
:1,$s/Ctrl-V-M//g
```

linux

replace

global - all occurances

substitute

text to match

1st line to start