• **Previous lecture:**
  – Structure & structure array

• **Today’s lecture:**
  – More on structs and cell array
  – Introduction to Matlab objects and classes

• **Announcements:**
  – **Project 5** due Friday 4/22 at 11pm
  – **Prelim 2** on Tues, 4/26, at 7:30pm
  – Prelim 2 topics: end with Project 5 and non-OOP part of this lecture, i.e., will NOT include OOP
Different kinds of abstraction

• Packaging procedures (program instructions) into a function
  – A program is a set of functions executed in the specified order
  – Data is passed to (and from) each function

• Packaging data into a structure
  – Elevates thinking
  – Reduces the number of variables being passed to and from functions
All possible (i,j,k) combinations but avoid duplicates. Loop index values have this relationship $i < j < k$

```
for i=1:n-2
    for j=i+1:n-1
        for k=j+1:n
            disp([i j k])
        end
    end
end
```
Still get the same result if all three loop indices end with \( n \)?

A: Yes
B: No

\[
\begin{array}{ccc}
1 & 2 & 3 \\
1 & 2 & 4 \\
1 & 2 & 5 \\
1 & 2 & 6 \\
1 & 3 & 4 \\
1 & 3 & 5 \\
1 & 3 & 6 \\
1 & 4 & 5 \\
1 & 4 & 6 \\
1 & 5 & 6 \\
\end{array}
\]

\[
\begin{array}{ccc}
2 & 3 & 4 \\
2 & 3 & 5 \\
2 & 3 & 6 \\
2 & 4 & 5 \\
2 & 4 & 6 \\
2 & 5 & 6 \\
\end{array}
\]

\[
\begin{array}{ccc}
3 & 4 & 5 \\
3 & 4 & 6 \\
3 & 5 & 6 \\
\end{array}
\]

\[
\begin{array}{c}
4 & 5 & 6 \\
i = 4
\end{array}
\]

\[
\begin{array}{c}
i = 3
\end{array}
\]

\[
\begin{array}{c}
i = 2
\end{array}
\]

\[
\begin{array}{c}
i = 1
\end{array}
\]

\[
\begin{array}{c}
\text{for } i=1:n \\
\text{for } j=i+1:n \\
\text{\qquad for } k=j+1:n \\
\text{\qquad \quad disp([i \ j \ k])} \\
\text{\quad end} \\
\text{end} \\
\text{end}
\end{array}
\]
Structures with array fields

Let’s develop a structure that can be used to represent a colored disk. It has four fields:

- \texttt{xc}: x-coordinate of center
- \texttt{yc}: y-coordinate of center
- \texttt{r}: radius
- \texttt{c}: rgb color vector

Examples:

\begin{verbatim}
D1 = struct('xc',1,'yc',2,'r',3,'c',[1 0 1]);
D2 = struct('xc',4,'yc',0,'r',1,'c',[.2 .5 .3]);
\end{verbatim}
Example: Averaging two disks

D1

D2
Example: Averaging two disks
Example: Averaging two disks
Example: compute “average” of two disks

% D1 and D2 are disk structures.
% Average is:

r = (D1.r + D2.r) /2;
xc = (D1.xc + D2.xc)/2;
yc = (D1.yc + D2.yc)/2;
c = (D1.c + D2.c) /2;

% The average is also a disk
D = struct(‘xc’,xc,’yc’,yc,’r’,r,’c’,c)
How do you assign to \( g \) the green-color component of disk \( D \)?

\[
D = \text{struct('xc',3.5, 'yc',2, ... 'r',1.0, 'c',[.4 .1 .5])}
\]

A: \( g = D.g; \)
B: \( g = D.c.g; \)
C: \( g = D.c.2; \)
D: \( g = D.c(2); \)
E: other
A structure’s field can hold a structure

\[
A = \text{MakePoint}(2,3) \\
B = \text{MakePoint}(4,5) \\
L = \text{struct}('P',A,'Q',B)
\]

- This could be used to represent a line segment with endpoints P and Q, for instance
- Given the MakePoint function to create a point structure, what is \( x \) below?

\[
x = L.P.y;
\]

A: 2  B: 3  C: 4  D: 5  E: error
Example: subset of clicker IDs


```
L = {}; k = 0;
for r = 1:size(IDs,1)
    if IDs(r,1) == 'h'
        k = k + 1;
        L{k} = IDs(r,:);
    end
end
```

Directly assign into a particular cell—good!

```matlab
L = {}; k = 0;
for r = 1:size(IDs,1)
    if IDs(r,1) == 'h'
        k = k + 1;
        L{k} = IDs(r,:);
    end
end
```

Concatenate cells or cell arrays—prone to problems!
End of coverage for Prelim 2
Different kinds of abstraction

- Packaging **procedures (program instructions)** into a function
  - A program is a set of functions executed in the specified order
  - Data is passed to (and from) each function

- Packaging **data** into a **structure**
  - Elevates thinking
  - Reduces the number of variables being passed to and from functions

- Packaging **data**, and the **instructions** that work on those data, into an **object**
  - A program is the interaction among objects
  - Object-oriented programming (OOP) focuses on the design of data-instructions groupings
A card game, developed in two ways

• Develop the algorithm—the logic—of the card game:
  – Set up a deck as an array of cards. (First, choose representation of cards.)
  – Shuffle the cards
  – Deal cards to players
  – Evaluate each player’s hand to determine winner

• Identify “objects” in the game and define each:
  – Card
    • Properties: suit, rank
    • Actions: compare, show
  – Deck
    • Property: array of Cards
    • Actions: shuffle, deal, get #cards left
  – Hand …
  – Player …

Procedural programming: focus on the algorithm, i.e., the procedures, necessary for solving a problem

• Then write the game—the algorithm—using objects of the above “classes”
A card game, developed in two ways

- Develop the algorithm—the logic—of the card game:
  - Set up a deck as an array of cards. (First, choose representation of cards.)
  - Shuffle the cards
  - Deal cards to players
  - Evaluate each player’s hand to determine winner

- Identify “objects” in the game and define each:
  - Card
    - Properties: suit, rank
    - Actions: compare, show
  - Deck
    - Property: array of Cards
    - Actions: shuffle, deal, get #cards left
  - Hand …
  - Player …

Object-oriented programming: focus on the design of the objects (data + actions) necessary for solving a problem

Procedural programming: focus on the algorithm, i.e., the procedures, necessary for solving a problem
Notice the two steps involved in OOP?

• Define the classes (of the objects)
  – Identify the properties (data) and actions (methods, i.e., functions) of each class

• Create the objects (from the classes) that are then used—that interact with one another
Defining a class ≠ creating an object

• A class is a specification
  – E.g., a cookie cutter specifies the shape of a cookie

• An object is a concrete instance of the class
  – Need to apply the cookie cutter to get a cookie (an instance, the object)
  – Many instances (cookies) can be made using the class (cookie cutter)
  – Instances do not interfere with one another. E.g., biting the head off one cookie doesn’t remove the heads of the other cookies