

- Review the code examples from class (found on the course website).
- Some parts below are associated with a question. **Answer the questions** using comments in your diary file.
- The office hours of the TAs and the instructor are on the class website. Visit us!

Problems

1. In a diary file, create a struct for a building that has properties for name, age, floors, and square footage.

Show how to change the value of the name of the building. Create a second struct for a building and then assign the first struct to be equal to the second struct.

Now change the name of the second struct. Display both structs again and **explain** in your diary file what you see (you can explain things in your diary file by writing them in a comment `>> %I see a building` or `>> %I see nothing`).

2. Create an array of building structs with different sizes for square footage. Create a function called **sizeLessThan** that compares two building structs and returns true if the first building is less than the size of the second building. Create a test function, **sizeLessThanTest**, that tests several cases.

```
>> building1.squareFeet = 5;
>> building2.squareFeet = 10;
>> sizeLessThan(building1, building2)
ans =
    true
```

3. Upgrade your buildings to a **Building** class definition. Create a

```
classdef
```

for **Building** and give it the same properties as before. Add a constructor function for **Building**.

In a diary file, create a vector of buildings using the new class constructor. Try calling your **sizeLessThan** function using these new building classes. Does this work? Explain why or why not.

4. Now create a parking lot class definition that only has two properties, spaces and squareFeetPerSpace.

Create a function called **getParkingLotSquareFeet** that calculates the total square feet for a given parking lot:

```
>> parkLot.squareFeetPerSpace = 100;
>> parkLot.spaces = 50;
>> getParkingLotSquareFeet(parkLot)
ans =
    5000
```

Write a test function that creates a parking lot and demonstrates your **getParkingLotSquareFeet** function working for parking lots.

5. Create a **getSquareFeet** function to work for both parking lot classes and for buildings using the

`isa`

function in MATLAB. The

`isa`

function takes an object and a classname as arguments:

```
isa(building1, 'Building') == true
```

Write a test function that creates a parking lot and a building and shows your **getSquareFeet** function working for both a building and a parking lot.

6. Write a function, **getSmallest** that finds and returns the smallest building or parking lot given a cell array of both buildings and parking lots. **getSmallest** should use your **getSquareFeet** function to determine which object is the smallest.

Write a test function that creates a cell array containing both parking lots and buildings and shows your **getSmallest** function working.

If your TA requires a paper copy, be sure that you have a printed copy of your function M-files, test function M-files, class definition M-files, and diary files demonstrating your testing. All must be stapled together, with your name and lab section on the top page.

Be sure that you upload a copy of all the MATLAB function, test functions, and diary files to Sakai. Then, click submit ONLY ONCE to send these to your Sakai and your TA.

On the first page of every printed copy for this course, your name, section, and TA's name must appear.