CMPE212 – Reminders

- Assignment 2 is posted. Due this Friday.
- Quiz 1 Grading Underway.

Today

- Start Encapsulation.

Encapsulation

- Encapsulation is the process of defining a Class that has at least one customizable attribute.
- Encapsulation is about the abstraction or containment of the attributes defined in the class.
- In Java, methods and attributes must be encapsulated or contained in a class definition.
- We have not yet built any classes with attributes that must be set upon instantiation. Time to start!

Encapsulation, Cont.

- The Best Object Design:
  - Supports the re-usability of code.
  - Provides a well-defined interface to other objects and the user(s).
  - Builds into an Object the code that ensures the integrity of the data stored in the Object by:
    - Making sure that initial data values are “legal”.
    - Controlling (or preventing) changes to data.
    - Preventing “privacy leaks” when data is returned out of an Object.
  - Works well with modular design practices making code easier to write, test and debug.

Encapsulation, Cont.

- This is the driving principle for the design of all Objects in the Java API!
  - So, maybe you should design your own Objects using the same principle?

An Example

- Consider the general model (or “pattern”…) of a database:
  - Collection
  - Database
  - User Interface
  - User

- Each record is an instance of an Object, too.
For now, we will focus on just how to define one of the many objects needed for the database—the object that will be used to define the record structure—a “Foundational” Object.

If each record only held one piece of data, we would not need an object for each record—the Collection could just be an array.

But—more typically—each database record will hold several pieces of data.

Suppose you want to create a database to store records of Halloween visits.

(Maybe we are going to build a model to predict the number of visitors for given weather conditions so we can predict how much candy to buy?)

For each year, you wish to record:

- The year.
- Number of munchkins banging at your door.
- Outdoor temperatures (one reading per hour) in deg C.
- Weather condition—“rain”, “snow”, or “clear”.

What would you do without objects and encapsulation?:

You could create four arrays, one for each of the fields.

- Difficult to code when moving records around.
- Suppose some record swap or delete is messed up and the records get out of sync—how would you know?

It would be easier to design a Class that contains (or encapsulates) these four fields and then design another class to collect instances of this Class.

- No way the fields could get out of sync since moving a record moves all fields at once.

public class Halloween1 {
    public int year;
    public int numMunchkins;
    public int[] temperatures;
    public String weatherCondition;
}

In a method in some other class:

Halloween1[] hwDB = new Halloween1[100];
 hwDB[0] = new Halloween1();
 hwDB[0].year = 1992;
 hwDB[0].numMunchkins = 200;
 hwDB[0].temperatures = [10, 9, 7, 6];
 hwDB[0].weatherCondition = "rain";
Halloween1, Cont.

• A question:

Why not declare the attributes in Halloween1 static?

Every instance of a class shares static members – there is only one copy of each static member in memory. If the attributes were static then each instance of Halloween1 would have the same values!

Halloween1, Cont.

• Another question:

What is wrong with the following?:

```java
hwDB[1] = new Halloween1();
hwDB[1].year = 3011;
hwDB[1].numMunchkins = -200;
int[] temp = {100, 200};
hwDB[1].temperature = temp;
hwDB[1].weatherCondition = "frogs";
```

Encapsulation, Cont.

• Attributes must be declared private, so that the class that owns them can control how they are set.

• If attributes are private then how can a user of the class assign the values of the attributes?

• Through methods, of course!

  - Constructor(s)
  - Mutator(s)

When instantiated only

Encapsulation, Cont.

• Within these methods, you can write code to check the parameters for validity.

  - What do you do if the values are not legal?

    - Throw an exception!
    - We need to start adding Behaviour to our Class.

Defining Exceptions

• You can throw an exception already defined in java, but:

  - Most of the time you will want to throw your own exception objects.

  - See the next slide for the definition of an Exception object.
Defining Exceptions, Cont.

```java
public class IllegalHalloweenException extends Exception {

    public IllegalHalloweenException() {
        super("Illegal parameter value supplied to Halloween object.");
    }

    public IllegalHalloweenException(String message) {
        super(message);
    }

} // end IllegalHalloweenException
```

Defining Exceptions, Cont.

- To be “throwable” this class must extend the class Exception.
- Otherwise this would not be an “exception” and could not be thrown or caught.
- The constructors in our class just use the constructors written in the Exception class.
- And other methods, like .getMessage() are inherited...

Defining Exceptions, Cont.

- Inside a method that detects an illegal parameter:
  ```java
  throw new IllegalHalloweenException("Illegal number of kids.");
  ```
- At the end of the method header that contains the above line of code:
  ```java
  throws IllegalHalloweenException {
  ```
- This part of the method header is called a “decoration”.

Defining Exceptions, Cont.

- This example contains a few Java keywords that we have not yet discussed:
  - `extends`
  - `super`
  - `throw`
  - `throws`
- The exact meaning and usage of `extends` and `super` will become clear when we discuss object hierarchies or inheritance.
- For now, just follow the pattern.

Assigning Private Attributes - Constructors

- Constructors are special methods that have the same name as the class in which they reside, but have no return type (not even `void`).
- They must be `public` to be used...
- They are only executed once, when the object is being instantiated. You can’t invoke them later.
- Constructors are often overloaded. This means you can have many constructors with different parameter lists. Why do that?

Halloween2

- Halloween2.java uses a constructor to set all parameters.
- It also throws an exception if illegal values are encountered.
- See how this object is used in TestHalloween2.java.
- Note that this class is not complete – it only has the one constructor, for now.
Aside - Information Privacy

- Having private attributes is a good start.
- But is temperatures (an array of int) really private?
- Can I change the contents of the private array from outside the class?
- How do I fix this?

What Does .clone() Do?

- Suppose you want to make a completely separate copy of an int array called stuff:
  ```java
  int[] clonedStuff = stuff.clone();
  ```
- Or:
  ```java
  int[] clonedStuff = new int[stuff.length];
  for(int i = 0; i < stuff.length; i++)
    clonedStuff[i] = stuff[i];
  ```
- Does the same thing.

More on .clone() (An array inherits its methods from the base class Object. Including a clone method that actually works for 1D arrays.)

- Every object in Java will have a clone method. But, they do not always work the way you would like.
- Our class should have a clone method that works!

Assigning Private Attributes - Constructors, Cont.

- Note that once you have written a constructor with parameters, you can no longer use a constructor without any parameters, called the default constructor.
- If you want an empty constructor, you must write one yourself, in addition to the parameterized constructor(s).
- Why would you want an empty constructor?

Aside – Preventing Instantiation

- Provide only the default constructor and make it private.
- Pretty sneaky!!
- Used by the Math class, for example.
- There is no need to instantiate the Math class since all its methods and attributes are static.

Assigning Private Attributes - Constructors, Cont.

- Suppose you don’t want to have to force the user to supply all parameters when he instantiates the object? How are you going to do this?
  - Overload the constructors.
- Suppose you want to edit parameters after you have created the object? How are you going to do this, too?
  - Provide mutator(s).
Assigning Private Attributes - Mutator Methods

• Called “set” methods – in fact, by convention, the word “set” is the first word in the method name.

• These methods must also check for legal parameters.
• Usually the constructor invokes mutators, when they exist.

• Should you write a mutator for each parameter?