Object Oriented Programming in Java
Monday, Week 3

OOP Concepts

- Last Week's Assignment
- Arrays
- Collection Class -- vector
- Threads
- Exception Handling
- Interface PinBallTarget

- Reading Budd, Ch 7, 18; AGH Ch. 3
- **Asst** (due Thursday)

  Ch. 7: Exercises 2,3,4 (5 optional), (8 everybody try!)

  8,9 (for intermediate and advanced --

  10 (for advanced)

  and any other pinball enhancement of your invention!

Arrays

```
private Ball [ ] ballArray;
private static final int BallArraySize = 10;

   ballArray = new Ball [ BallArraySize ];
   for (int i = 0; i < BallArraySize; i++) {
       ballArray[i]  = new Ball (10,15,5);
       ballArray[i].setColor (ballColor);
       ballArray[i].setMotion (3.0+i, 6.0-i);
   }
```

Arrays

- Creating an array does not create the objects that are stored in the array.
- Two ways to create Java arrays
  - 1. Button buttons[ ] = new Button[10];
  - 2. int lookup_table[ ] = {1, 2, 4, 8, 16};

    Menu m = createMenu("File", new String[ ]

    {"Open…", "Save", "Quit"});
- The size of an array is not part of its type

```
String[ ] strings;
strings = new String[10];
strings = new String[20];
```

- Arrays behave "like" objects, e.g.,

```
x = ballArray.length
```

Collection Classes

`(set, list, vector, ...)`

- Collections (aka containers) are holders that letyou store and organize objects in useful ways for efficient access.
- java.util contains the general collection framework.
- java.lang.Comparable

<u>Iteration</u>

```
public boolean hasNext ()
public Object next()
public void remove()
public void add (Object elem)
public void set (Object elem)
```

```
public int size()
public boolean isEmpty()
public boolean contains
       (Object elem)
public boolean remove
       (Object elem)
```

<u>Ordering (using Comparable and Comparator)</u>

```
public int compareTo (Object other)
public int compare (Object o1, Object o2)
```

Collection Classes

The legacy collection types: **Vector**, Stack, etc.

- Vector. All methods that access the contents of a Vector are synchronized. Vector is analogous to ArrayList, and so inherits from List.
- How are arrays and vectors alike, and how different?

```
public final void addElement (Object elem)
public final void insetElementAt(Object elem, int index)
public final void removeElementAt(int index)
public final void removeAllElements ( )
```

Last week’s homework….

- If not finished, do that before going on…
- if not happy with it - you will have a chance to redo (& resubmit) BEFORE the midterm

- speaking of the midterm….
    - Ok to put off til wed of week 6?
    - Short answer (15 min) -- week 5?

- Robert Murphy????

This Week's Lab & Homework

- Norman's two kinds of cognition --
  - *experiential* and *what???*
- Advice
  - make yourself a map of the program space!
  - Think how this might be different programming for you….
    - use the API ref on help page or http://grace/jdk_1_3_1/index.html the java API docs
  - have a plan before you jump in!
- become better acquainted with the tool
  - be aware of differences in browsers…
    - PDF doc -- Integrated Development Environment

Threads

- Threads are independent parts of a program that run simultaneously.
- Except on machines with more than one processor, only one thread is actually running at a time.
- Threaded programs are managed by a process that deciddes which thread runs at any given time.
- You can’t use them to sew on buttons…

Threads in Java

- Java is one of only a few programming languages that support multiple threads.
- The implementation of threads in Java differs among platforms.
- The differences can cause surprises if you aren’t aware of the threaded nature of most Java programs.

Concurrent Programming

- Multiple threads create many problems related to the order in which events occur.
- For example, if two threads access the same data field of an object, the results can depend on the timing of the access.
- Access to shared data must be synchronized in order to avoid unpredictable, incorrect results.

Concurrency Example

- Consider the following two code fragments that are executing concurrently, unsynchronized, and accessing the variables a,b,c, from the same object. Assume b=3 and c=2, initially:

Thread 1
```
a = b + 10;
c = a + 5;
```

Thread 2
```
b = c - 4;
```

What is the final value of a?
of b?
of c?

Synchronization

- Java provides synchronization so that situations such as the example don’t occur.
- Code that accesses shared data is kinown as a critical section.
- Critical sections are synchronized by obtaining exclusive access to the data they modify.

Synchronization Example

- With synchronization, only two orders of execution are possible in our example.
  - S1, s2, s3
  - s3, s1, s2
- The third possible ordering of the statements is presented by the synchronization.
- The possible orders produce different values, but each is a predicatable result that doesn’t depend on a so-called “race condition”.

Control of Threads

- Java provides methods that allow:
  - Suspension of a thread
  - Termination of a thread
  - Creation of new threads
  - Synchronization of threads

Creation of Threads

- There are two ways to define classes that can run as separate threads:
  - extending the class Thread
  - implementing the Runnable interface.
- Which you choose depends on the situation
  - Implementing Runnable makes sense if you want to inherit from a class that is not a subclass of Thread.

Exception Handling

Interfaces -- PinBallTarget

- 3 domain classes -- Peg, Wall, Hole and Spring share the same behavior (type), but have no structure in common.
- The behavior may be implemented differently.
- A variable can be declared to hold a value of PinBallTarget type (and thus a value of either Peg, Wall or Spring)