Recitation 5

 Enums and
 The Java Collections classes/interfaces
How do we represent . . .

- Suits - Clubs, Spades, Diamonds, Hearts
- Directions - North, South, East, West
- Days of week - Monday, Tuesday . . .
- Planets - Mercury, Venus, Earth . . .

Other small sets of values that do not change
Using constants

```java
public class Suit {
    public static final int CLUBS = 0;
    public static final int SPADES = 1;
    public static final int DIAMONDS = 2;
    public static final int HEARTS = 3;
}
```

Problems:
- no type checking
- readability

```java
void setSuit(int suit) {...}
int getSuit() {...}
```
Objects as constants

```java
public class Suit {
    public static final Suit CLUBS = new Suit();
    public static final Suit SPADES = new Suit();
    public static final Suit DIAMONDS = new Suit();
    public static final Suit HEARTS = new Suit();

    private Suit() {}  // cannot modify Suit objects
}
```

no new Suits can be created

Suit v;  …  if (v == Suit.CLUBS) { … }  use ==
Enum declaration

could be any access modifier

```java
public enum Suit {CLUBS, SPADES, DIAMONDS, HEARTS};
```

new keyword

name of enum

static final variables

of enum Suit
About enums

1. Can contain methods, fields, constructors
   a. `Suit.HEARTS.getColor();`

1. Suit’s constructor is private!
   a. Cannot instantiate except for initial constants

1. `Suit.values()` returns a `Suit[]` of constants in enum
Create a class PlayingCard and class Deck. What would be the fields for a PlayingCard object?
Enum odds and ends

1. Suit is a subclass of `java.lang.Enum`

2. `ordinal()` returns position in list (i.e. the order it was declared)
   a. `Suit.CLUBS.ordinal() == 0`

3. Enums automatically implement Comparable
   a. `Suit.CLUBS.compareTo(Suit.HEARTS)` uses the ordinals for Clubs and Hearts

4. `toString()` of `Suit.CLUBS` is “CLUBS”
   a. you can override this!
5. **switch** statement

```java
Suit s = Suit.CLUBS;
switch (s) {
    case CLUBS: 
    case SPADES: 
        color = "black"; break;
    case DIAMONDS: 
    case HEARTS: 
        color = "red"; break;
}
```

- `s == Suit.CLUBS` is true
- `switch` statements are fall through!
- `break` keyword is necessary.
The Collections classes and interfaces are designed to provide implementations of

- bags (like a bag of objects with duplicates allowed)
- sets
- lists
- Stacks
- queues

You will see in later assignments how easy it is to use these
Power of inheritance and interfaces

Format of ArrayList object

Collections and Map
Important interfaces

Collection<E>
- add(E);
- contains(Object);
- isEmpty();
- remove(Object);
- size();
- ...

Map<K,V>
- put(K,V);
- get(Object);

No new methods in Set<E>, just changes specifications

List<E>
- get(int);
- indexOf(int);
- add(int,E);
- ...

Set<E>
Queues? Stacks?

```
Collection<E>
  
Queue<E>
  
Deque<E>
  
LinkedList<E>  ArrayDeque<E>
```

“Double Ended Queue”
Iterating over a HashSet or ArrayList

HashSet s = new HashSet();

... store values in the set ...

for (Object e : s) {
    System.out.println(c);
}

Body of loop is executed once with e being each element of the set. Don’t know order in which set elements are processed
Collections problems

1. Remove duplicates from an array
2. Find all negative numbers in array
3. Create ransom note
4. Implement a Stack with a max API
5. Braces parsing
Collections problems

Complete
Integer[] removeDuplicates(int[])

Remove all duplicates from an array of integers.

Very useful HashSet method: 
hs.toArray(new Integer[hs.size()]);
Collections problems

Find Negative Numbers
Find all negative numbers in array and return an array with those integers

Very useful ArrayList method:

```
lst.toArray(new Integer[lst.size()]);
```
Create Ransom Note

Given a note (String) that you would like to create and a magazine (String), return whether you can create your note from the magazine letters.
Collections problems

Implement a Stack\(<E>\) with a max() function in O(1) time

No matter how full the stack is, the max function should be in constant time. (ie you should not iterate through the Linked List to find the maximum element)
Braces parsing in O(n) time

Return whether a String has the right format of square brackets and parenthesis.

e.g.
“array[4] = (( ( new Integer(3) ) ) );” <- is true
“(   ) [   ] ” <- is false
“ ) (” <- is false
“ ( [ ] ) ” <- is false
Collections problems

Print a binary tree in level-order

Output: 1 2 3 4 5 6

Challenge Problem
Output: 
1 
2 3 
4 5 6