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() {}  
}
```

```java
Suit v;   ...   if (v == Suit.CLUBS) { ...}  
```

Enum declaration

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

About enums

1. Can contain methods, fields, constructors
   - `Suit.HEARTS.getColor();`
2. Suit's constructor is private!
   - Cannot instantiate except for initial constants
3. `Suit.values()` returns a `Suit[]` of constants in enum
Demo: Enums in action

Look at the Suit enum.

Create a class PlayingCard and a 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!

Collections and Maps

The Collections classes and interfaces are designed to provide implementations of

- bags (a.k.a. multiset – sets with repeated values)
- sets (and sorted sets)
- lists
- stacks
- queues
- maps (and sorted maps)

You will see in later assignments how easy it is to use these

Power of inheritance and interfaces

Object

AbstractCollection<E>

AbstractList<E>

ArrayList<E>

Format of ArrayList object

Iterable<E>

Collection<E>

List<E>

Important interfaces

Collection<E>

add(E);
contains(Object);
isEmpty();
remove(Object);
size(); ...

Map<K,V>

put(K,V);
get(Object);

List<E>

get(int);
indexOf(int);
add(int,E);
...

Set<E>

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

- Collection<E>
- Map<K,V>
- Set<E>
- List<E>
- LinkedList<E>
- HashSet<E>
- ArrayList<E>
- HashMap<K,V>

### Queues? Stacks?

- Collection<E>
- Queue<E>
- Deque<E>

### 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

### Iterating over a HashSet or ArrayList

```java
HashSet<E> s = new HashSet<E>();
... store values in the set ...
for (E e : s) {
    System.out.println(e);
}

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

#### 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()]);
```
Collections problems

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.

```
give me the money or java gets it.
```

Collections problems

Implement a Stack<\text{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)

Collections problems

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
```

Collections problems

Print a binary tree in level-order
Challenger Problem
```
Output: 1 2 3 4 5 6
```
```
1
 /   \
2   3
 /   /
4   5 6
```