

## Recitation 5

Enums and  
The Java Collections classes/interfaces

1

## How do we represent . . .

Enums

- 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

2

## Using constants

Enums

```
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

```
void setSuit(int suit) {...}
int getSuit() {...}
```

3

## Objects as constants

Enums

```
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 == Sut.CLUBS){ ...} use ==
```

4

## Enum declaration

Enums

could be any access modifier

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

new keyword

name of enum

static final variables  
of enum Suit

5

## About enums

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

6

Enums

## Demo: Enums in action

---

Create a class PlayingCard and class Deck.  
 What would be the fields for a PlayingCard object?

7

Enums

## 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!

8

Enums

## Enum odds and ends

---

5. **switch** statement

```
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.

9

Collections and Map

## Collections and Map

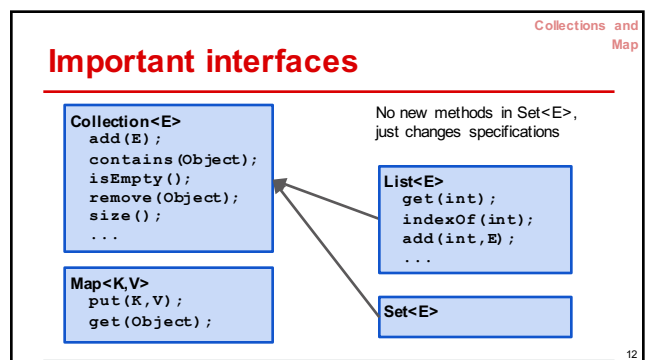
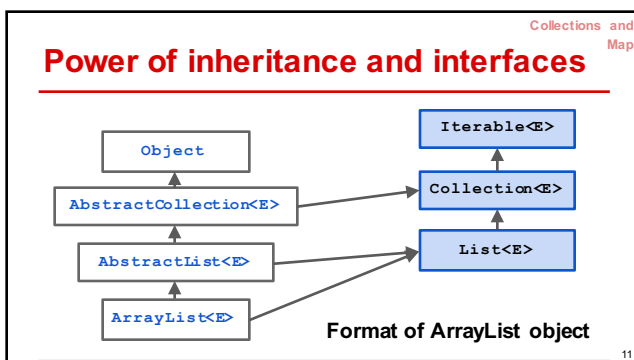
---

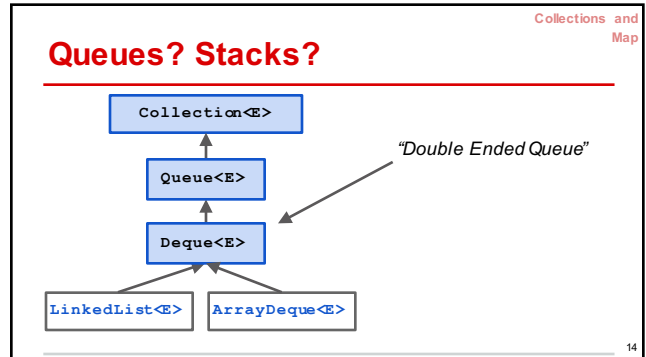
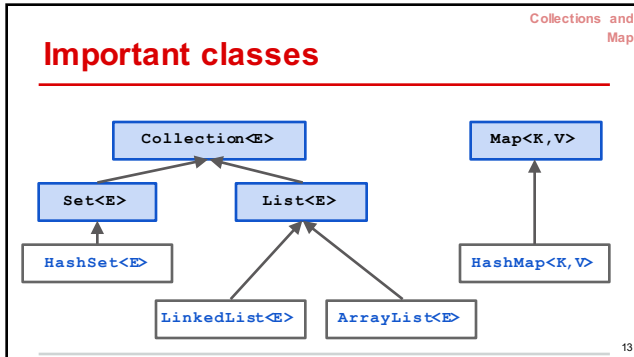
The Collections classes and interfaces are designed to provide implementations of

- 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

10





### 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

HashSet@y2

Object  
HashSet

Fields contain a set of objects

HashSet() add(Object)  
contains(Object) size()  
remove(Object) ...

s HashSet@y2 HashSet

- ### 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()]);
  
```

Collections and Map

## 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.

g I V E M E U / E  
 M O n E Y O R J A V A  
 g E U S I U .

19

Collections and Map

## 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)

20

Collections and Map

## 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
" ( [ ] ) " <- is false
    
```

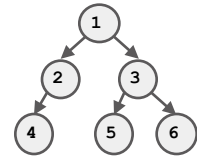
21

Collections and Map

## 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

22