CS210 Discussion

Week 6

Project 3 – Comparisons

- Comparisons
  - Comparable
  - Comparator
- Binary Search

- Term
- Binary Search Deluxe
- Autocomplete

Comparable vs. Comparator

Comparable

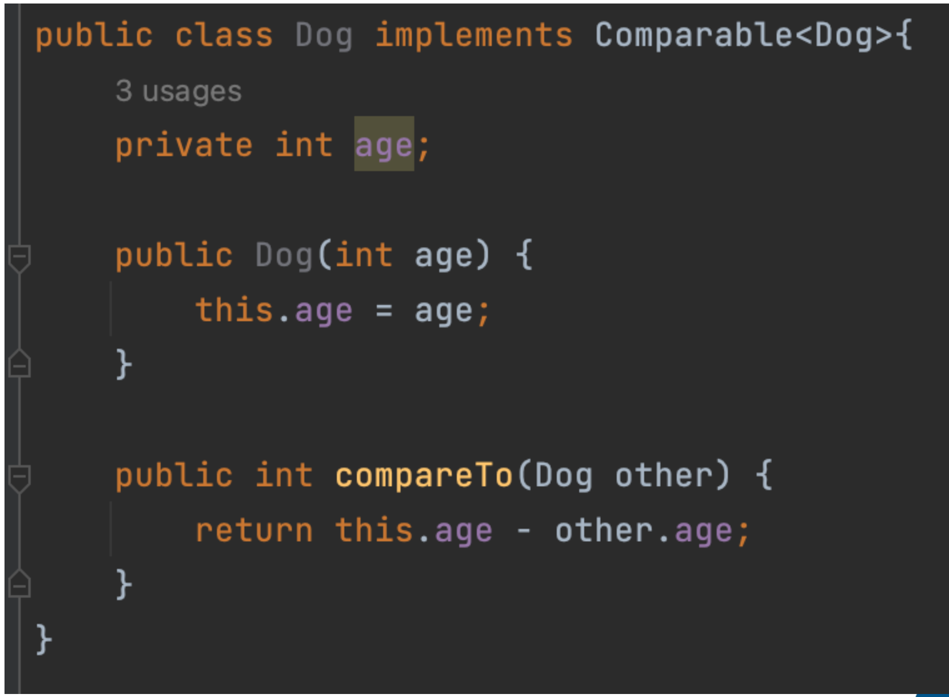
- compareTo()
- An instance of the thing you're comparing
- Typically represents a 'natural' order

Comparator

- compare()
- A separate object who's purpose is to compare two others
- Typically represents an alternative order
- Must be created before it can be used

Comparable AND Comparator

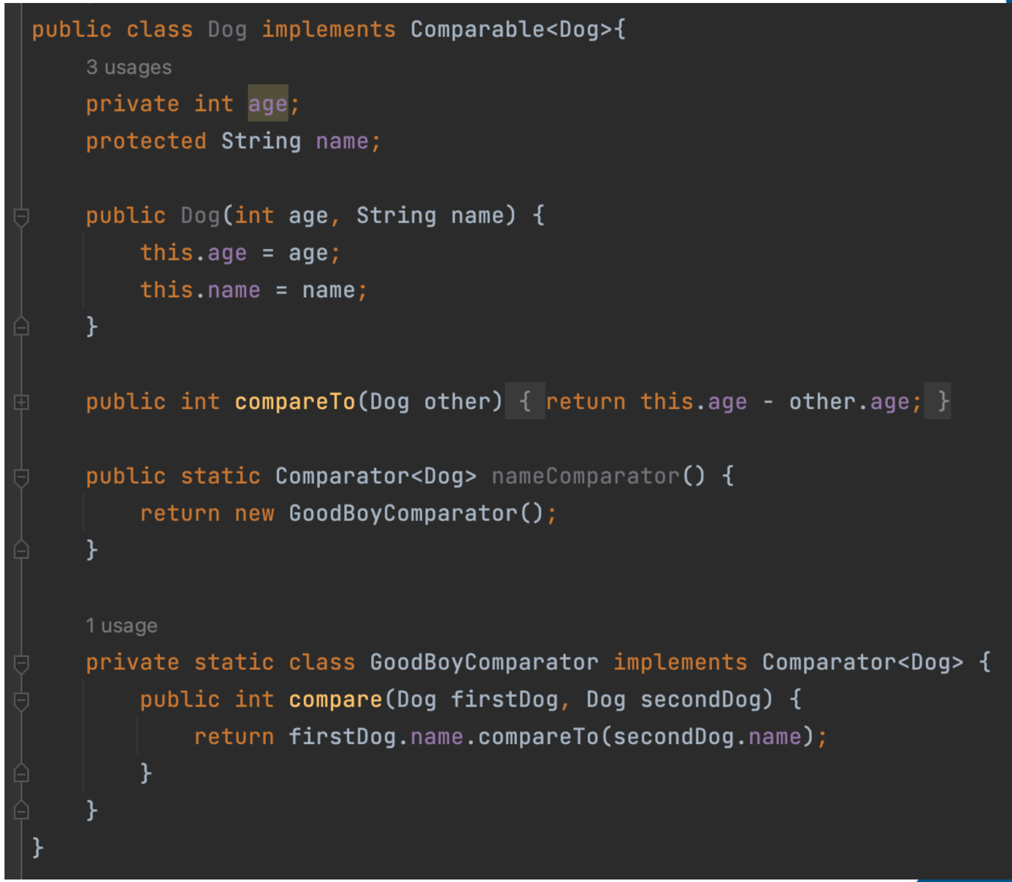
- Comparison results are integers
  - < 0 if the first is smaller than the second
  - 0 if they're equal
  - > 0 if the first is bigger than the second
- For comparable, the first item is the object itself



```
public class Dog implements Comparable<Dog>{
    3 usages
    private int age;

    public Dog(int age) {
        this.age = age;
    }

    public int compareTo(Dog other) {
        return this.age - other.age;
    }
}
```

Comparable AND Comparator

- Comparison results are integers
  - < 0 if the first is smaller than the second
  - 0 if they're equal
  - > 0 if the first is bigger than the second
- For comparable, the first item is the object itself



```java
public class Dog implements Comparable<Dog>{
    3 usages
    private int age;
    protected String name;

    public Dog(int age, String name) {
        this.age = age;
        this.name = name;
    }

    public int compareTo(Dog other) { return this.age - other.age; }

    public static Comparator<Dog> nameComparator() {
        return new GoodBoyComparator();
    }

    1 usage
    private static class GoodBoyComparator implements Comparator<Dog> {
        public int compare(Dog firstDog, Dog secondDog) {
            return firstDog.name.compareTo(secondDog.name);
        }
    }
}
```

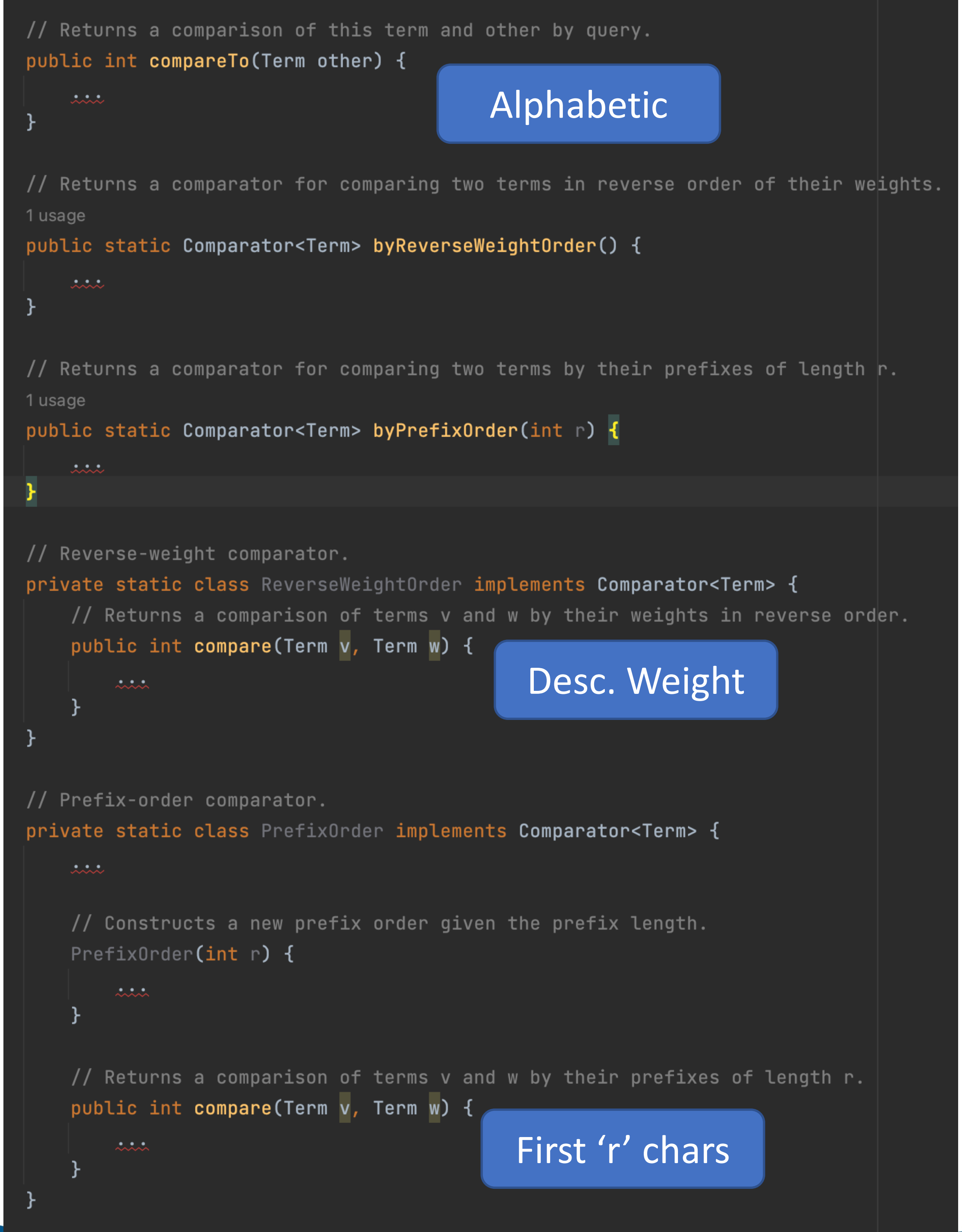
Problems

- Goal is to build an autocomplete engine
- First a Term object
- Then a BinarySearchDeluxe algorithm
- Finally the Autocomplete itself, putting them together

Term

- An autocomplete word
  - The word itself
  - It's weight
- Comparable
  - Lexicographic == alphabetic order
- Two additional comparators
  - Desc. by weight
  - Lexi. for first 'r' chars

P1



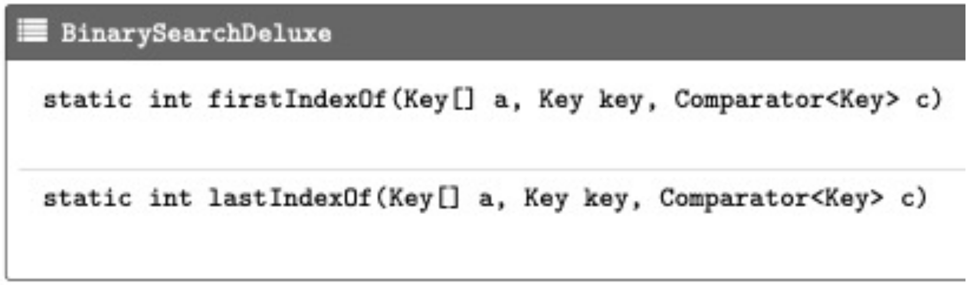
```
// Returns a comparison of this term and other by query.
public int compareTo(Term other) {
    ...
}

// Returns a comparator for comparing two terms in reverse order of their weights.
1 usage
public static Comparator<Term> byReverseWeightOrder() {
    ...
}

// Returns a comparator for comparing two terms by their prefixes of length r.
1 usage
public static Comparator<Term> byPrefixOrder(int r) {
    ...
}

// Reverse-weight comparator.
private static class ReverseWeightOrder implements Comparator<Term> {
    // Returns a comparison of terms v and w by their weights in reverse order.
    public int compare(Term v, Term w) {
        ...
    }
}

// Prefix-order comparator.
private static class PrefixOrder implements Comparator<Term> {
    ...

    // Constructs a new prefix order given the prefix length.
    PrefixOrder(int r) {
        ...
    }

    // Returns a comparison of terms v and w by their prefixes of length r.
    public int compare(Term v, Term w) {
        ...
    }
}
```

Term

- Comparison methods should run in T(n) ~ n
- Length of the string
  - Chars to resolve comparison

Binary Search Deluxe

- Like normal binary search but instead of one index, give two
  - A range of the 'same' element
- Uses Terms build in P1
- Terms have comparators
- Runs in T(n) ~ log n



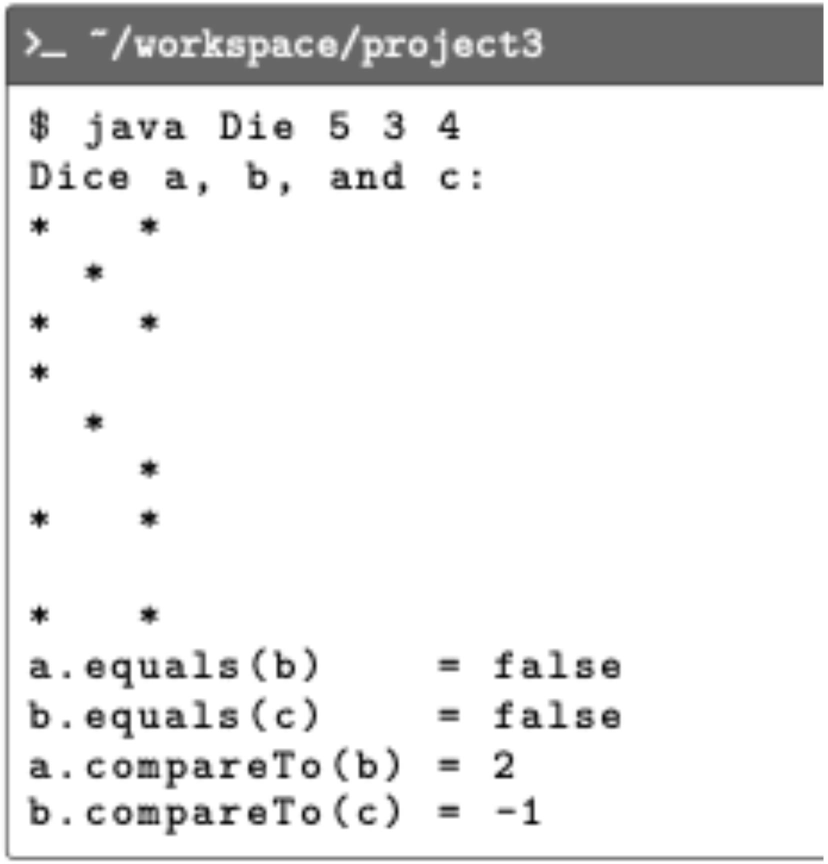
BinarySearchDeluxe

```
static int firstIndexOf(Key[] a, Key key, Comparator<Key> c)

static int lastIndexOf(Key[] a, Key key, Comparator<Key> c)
```

Questions about the first two problems?

E1

Six-Sided Dice



```
>_ ~/workspace/project3
```

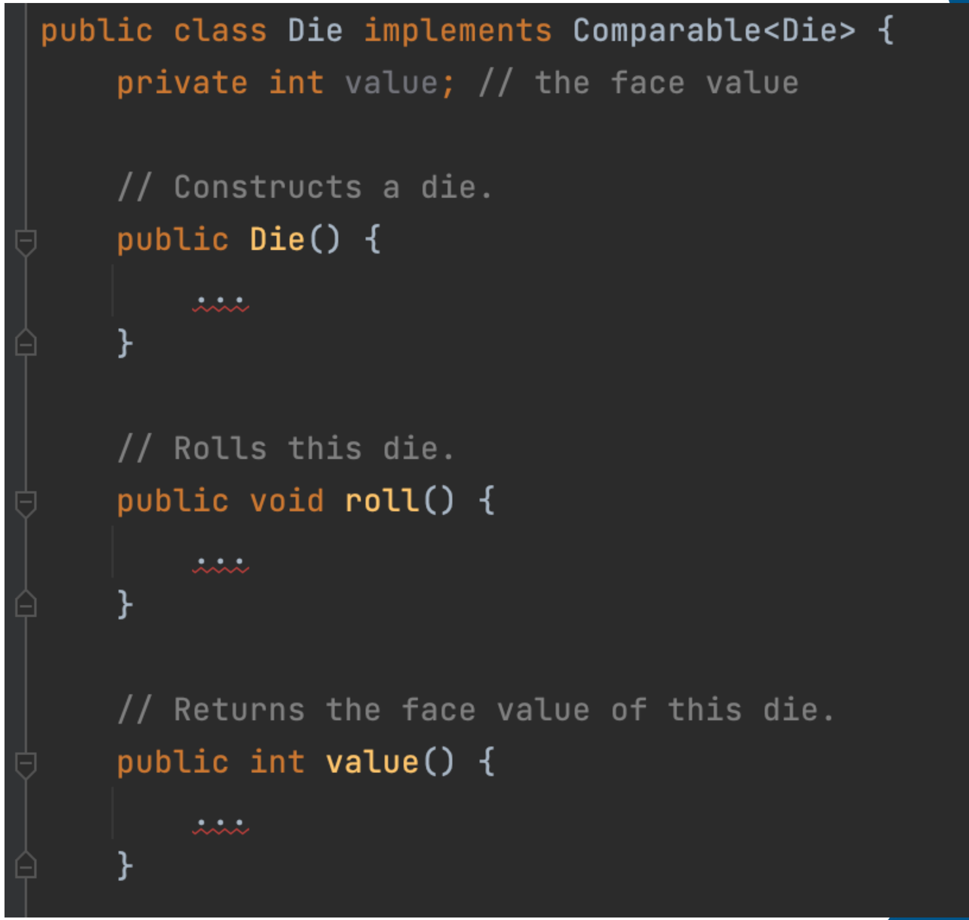
```
$ java Die 5 3 4
Dice a, b, and c:
*   *
  *
*   *
*
  *
    *
*   *

*   *
a.equals(b)    = false
b.equals(c)    = false
a.compareTo(b) = 2
b.compareTo(c) = -1
```

- A Die object to represent the roll of a die
- Can print the die face
- Is comparable based on face value

E1

Six-Sided Dice

- Simple constructor
- Roll needs to change the die's value to a random number [0, 6]
- Value is a simple getter

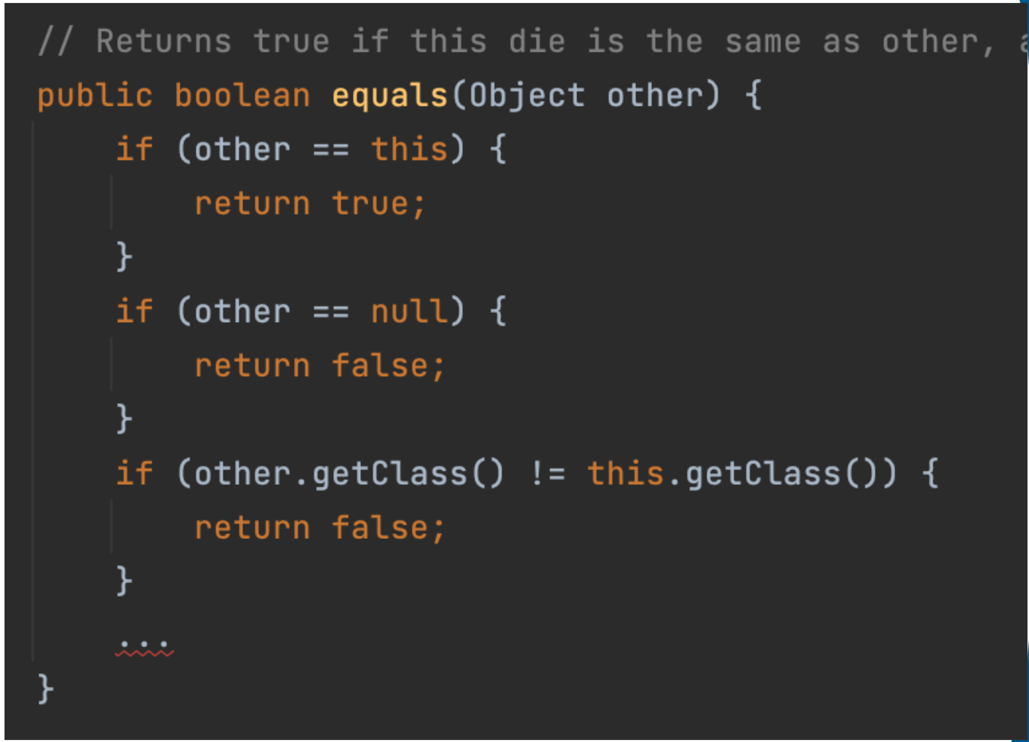


```java
public class Die implements Comparable<Die> {
    private int value; // the face value

    // Constructs a die.
    public Die() {
        ...
    }

    // Rolls this die.
    public void roll() {
        ...
    }

    // Returns the face value of this die.
    public int value() {
        ...
    }
```

E1

Six-Sided Dice

- 'equals' may look familiar
- Comparing die by face value
  - What's the syntax?
- Returns a Boolean

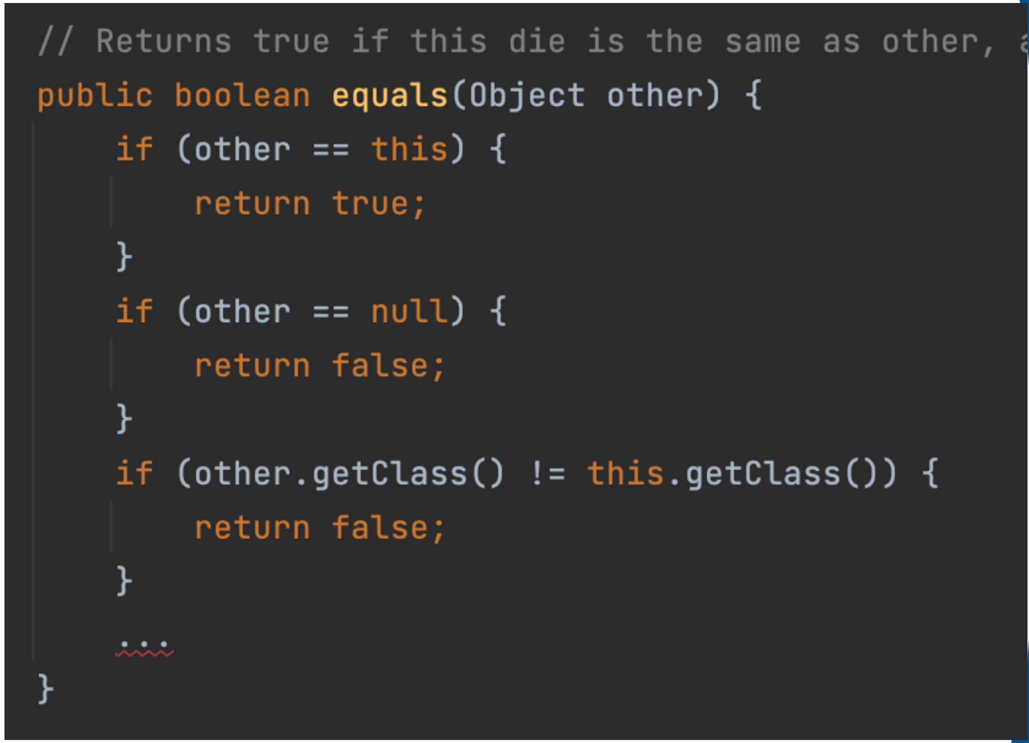


```
// Returns true if this die is the same as other, a
public boolean equals(Object other) {
    if (other == this) {
        return true;
    }
    if (other == null) {
        return false;
    }
    if (other.getClass() != this.getClass()) {
        return false;
    }
    ...
}
```

E1

Six-Sided Dice

- 'equals' may look familiar
- Need to cast 'other' to Die
  - What's the syntax?
- Comparing die by face value
  - What's the syntax?
- Returns a Boolean

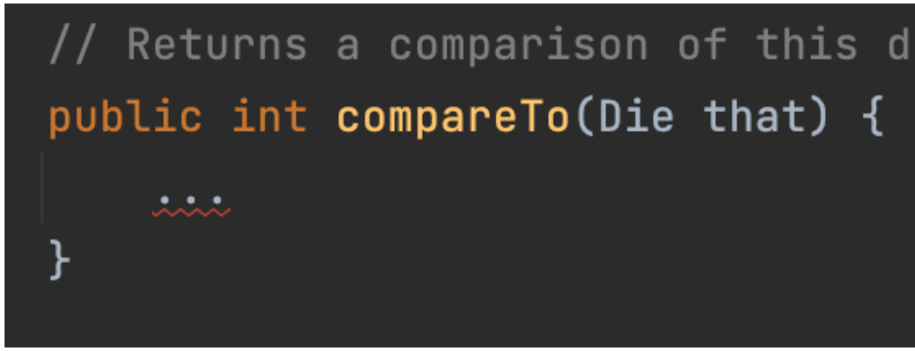


```
// Returns true if this die is the same as other, a
public boolean equals(Object other) {
    if (other == this) {
        return true;
    }
    if (other == null) {
        return false;
    }
    if (other.getClass() != this.getClass()) {
        return false;
    }
    ...
}
```

E1

Six-Sided Dice

- We need to return an integer
  - < 0 if 'this' value is less than 'that' value
  - 0 if equal
  - > 0 if 'this' value is greater than 'that' value
- How can we do this?
  - What can we write to do this in one line?



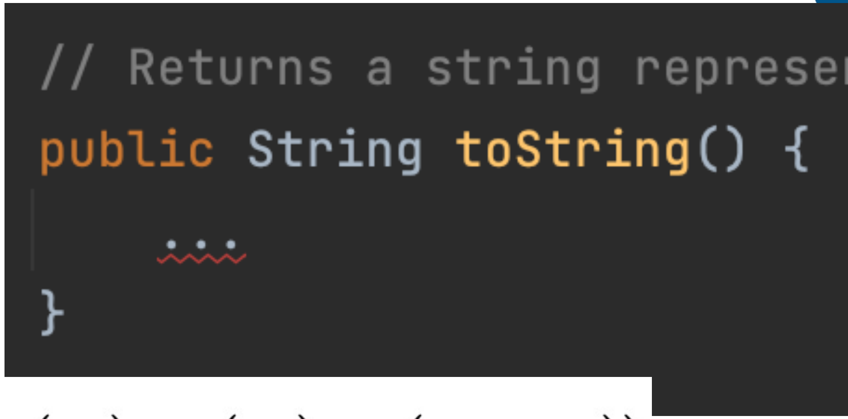
```
// Returns a comparison of this d
public int compareTo(Die that) {
    ...
}
```

UMass Boston

E1

Six-Sided Dice

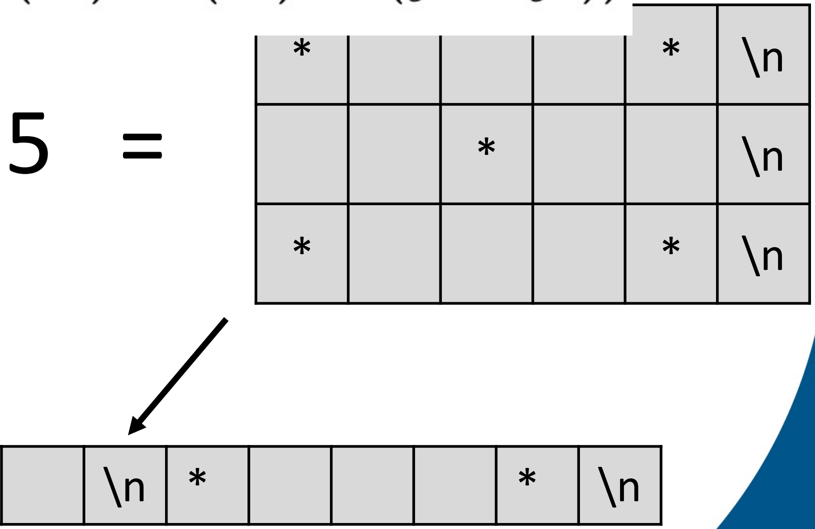
- StringBuilder
  - append()
  - toString()



```
// Returns a string represe
public String toString() {
    ...
}
```

$$d = 6359.83 \arccos(\sin(x_1)\sin(x_2) + \cos(x_1)\cos(x_2)\cos(y_1 - y_2)).$$

- Case statement
  - 1 for each number 1 – 6



5 =

*				*	\n
		*			\n
*				*	\n

*				*	\n			*			\n	*				*	\n

E2

Location

- Similar to Die
- Work on your own or in small groups, 10 min
- ‘equal’ needs to check all three: name, lat, and lon
- ‘compareTo’ is based on great circle distance to Greece

$$d = 6359.83 \arccos(\sin(x_1)\sin(x_2) + \cos(x_1)\cos(x_2)\cos(y_1 - y_2)).$$

UMass Boston

Questions?