

hashCode and equals

# Object methods

- Every object in Java must implement various methods, including:
  - `public boolean equals(Object x)`
  - `public int hashCode()`
  - `public String toString()`

# Implementing equals

- The signature of *equals* in Java is:

*boolean equals(Object x)*

- When implementing *equals*, we need to check for the equality each field which forms part of the “primary key” of an object. If any pair of fields is unequal, then the objects are unequal.
- Before we can compare the fields, we must establish that both objects have the same class otherwise it makes no sense to talk about comparing fields.
- And before doing that we might as well check a couple of other things that can give us an immediate result.

# Actual code: *equals*

- Example: java.lang.String:

```
public boolean equals(Object anObject) {  
    if (this == anObject) return true;  
    if (anObject instanceof String) {  
        String anotherString = (String)anObject;  
        int n = value.length;  
        if (n == anotherString.value.length) {  
            char v1[] = value;  
            char v2[] = anotherString.value;  
            int i = 0;  
            while (n-- != 0) {  
                if (v1[i] != v2[i])  
                    return false;  
                i++;  
            }  
            return true;  
        }  
    }  
    return false;  
}
```

- Example: java.time.LocalDate:

```
public boolean equals(Object obj) {  
    if (this == obj) return true;  
    if (obj instanceof LocalDate) return compareTo((LocalDate) obj) == 0;  
    return false;  
}
```

# What is a hash code?

- A hash code is a 32-bit *digest* of an object.
- A hash code should distribute all possible values of the object *uniformly* among all 4 billion possible values (the intention is to reduce the number of *collisions*: different objects, same hash).
- It is required to be consistent with *equals* such that:
  - *if  $a.equals(b)$  then  $a.hashCode == b.hashCode$*
  - It also follows that: *if  $a.hashCode \neq b.hashCode$  then  $! a.equals(b)$*

# Implementing hashCode

- It stands to reason, then, that the fields of a class that are tested in *equals* must also contribute to *hashCode*, otherwise the contract cannot be maintained.
- So, how do fields contribute to *hashCode*?
  - Typically, we calculate the *hashCode* of a field by calling *hashCode* on it (or on the boxed version of it if the field is a non-*int* primitive);
  - Once we have the various field *hashCode* values, we typically combine them together by some formula involving prime numbers such as:
    - $H = h_1 * p_1 + h_2 * p_2 + \dots + h_n * p_n$
    - In practice, the standard way to implement *hashCode* in Java is (in this example,  $n=4$ ):
    - $H = 31 * (31 * (31 * h_4 + h_3) + h_2) + h_1$

# The actual code

- Example: java.lang.String:

```
public int hashCode() {  
    int h = hash; // cached value: defaults to 0  
    if (h == 0 && value.length > 0) {  
        char val[] = value;  
        for (int i = 0; i < value.length; i++) {  
            h = 31 * h + val[i];  
        }  
        hash = h;  
    }  
    return h;  
}
```

- Example: java.time.LocalDate:

```
public int hashCode() {  
    int yearValue = year;  
    int monthValue = month;  
    int dayValue = day;  
    return (yearValue & 0xFFFF800) ^ ((yearValue << 11) + (monthValue << 6) + (dayValue));  
}
```

# Actual code continued

- Example: edu.neu.coe.info6205.bqs.Element\*:

```
@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null || getClass() != o.getClass()) return false;
    Element<?> element = (Element<?>) o;
    return Objects.equals(item, element.item) &&
        Objects.equals(next, element.next);
}

@Override
public int hashCode() {
    return Objects.hash(item, next);
}
```

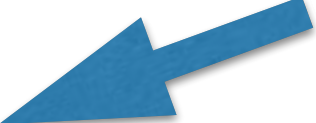
\* auto-generated by IDE



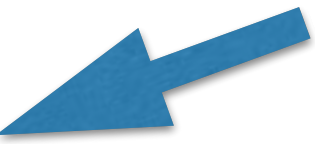
# Actual code: MyDate

```
package edu.neu.coe.info6205;
import java.util.Objects;
public class MyDate implements Comparable<MyDate> {
    public MyDate(int year, int month, int day) {
        this.year = year;
        this.month = month;
        this.day = day;
    }
    @Override public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        MyDate myDate = (MyDate) o;
        return year == myDate.year &&
            month == myDate.month &&
            day == myDate.day;
    }
    @Override public int hashCode() {
        return Objects.hash(year, month, day);
    }
    public int getDayOfWeek() {
        if (dayOfWeek == -1)
            dayOfWeek = java.time.LocalDate.of(year, month, day).getDayOfWeek().getValue();
        return dayOfWeek;
    }
    @Override public int compareTo(MyDate that) {
        int cfy = Integer.compare(this.year, that.year);
        if (cfy != 0) return cfy;
        int cfm = Integer.compare(this.month, that.month);
        if (cfm != 0) return cfm;
        return Integer.compare(this.day, that.day);
    }
    public int getYear() {
        return year;
    }
    public int getMonth() {
        return month;
    }
    public int getDay() {
        return day;
    }
    private final int year;
    private final int month;
    private final int day;
    private transient int dayOfWeek = -1;
}
```

*equals* and *hashCode* were auto-generated by IntelliJ IDEA.



*dayOfWeek* field is marked transient which implies that it is not part of Primary Key and therefore not serialized nor used in equals/hashCode.



# Equable

- Wouldn't it be nice if Java actually made it possible to ensure consistency between *hashCode* and *equals*?
- Please see the Equable class in the class repository and, for an example of its use, see ComparableTuple.
- Note that this last also ensures consistency with *Comparable*.