Announcements

- A3 now available on CMS and Piazza. Refer often to the Piazza FAQ Note for A3
- Please read the assignment FAQ Notes on the Piazza before asking a question. It might already be answered.

Assignment A3: Doubly linked Lists

Idea: maintain a list (2, 5, 7) like this:

This is a singly linked list

To save space we write names like a6 instead of N@35abcd00

How to insert a node at the beginning

How to remove a node from the middle

Assignment A3: Use an inner class

```java
public class LinkedList {
    private int x;
    public void m(int y) { ... }
    private class CI {
    }
}
```

Inside-out rule: Objects of CI can reference components of the object of C in which they live.

In addition, methods of C can reference private components of CI
Assignment A3: Generics

```java
public class LinkedList {
    void add(Object elem) {…}
    Object get(int index) {…}
}

Values of linked list are probably of class Object

public class LinkedList<E> {
    void add(E elem) {…}
    E get(int index) {…}
}

You can specify what type of values

ns = new LinkedList<Integer>();
ns.add("Hello"); // error
ns.add(5);
String s = ns.get(0); // error
int n = ns.get(0);

ss = new LinkedList<String>();
ss.add("Hello");
ss.add(5); // error
String s = ss.get(0);
int n = ss.get(0); // error
```

Overview ref in text and JavaSummary.pptx

- Quick look at arrays slide 50-55
- Casting among classes C.33-C.36 (not good) slide 34-41
- Consequences of the class type slide 34-41
- Operator instanceof slide 40
- Function equals slide 37-41

Homework. Learn about while/ for loops in Java. Look in text.

```java
while (<bool expr>) { … } // syntax
for (int k= 0; k < 200; k= k+1) { … } // example
```

Classes we work with today
class hierarchy:

Work with a class Animal and subclasses like Cat and Dog
Put components common to animals in Animal
Object partition is there but not shown

Bottom-up or overriding rule says function toString in Cat

Which function is called?

```java
Which function is called by 
v[0].toString() ?
```

Bottom-up or overriding rule says function toString in Cat
Consequences of a class type

Animal[] v; declaration of v. Also means that each variable v[k] is of type Animal.

The type of v is Animal]

The type of each v[k] is Animal

The type is part of the syntax/grammar of the language. Known at compile time.

As we see on next slide, the type of a class variable like v[k] determines what methods can be called.

From an Animal variable, can use only methods available in class Animal

a.getWeight() is obviously illegal. The class won’t compile.

When checking legality of a call like a.getWeight(...) since the type of a is Animal, function getWeight must be declared in Animal or one of its superclasses.

Another example

Type of v[0]: Animal

Should this call be allowed? Should program compile?

v[0].getWeight()
Each element \( v[k] \) is of type \( \text{Animal} \). From \( v[k] \), see only what is in partition \( \text{Animal} \) and partitions above it.

Components are in lower partitions, but can’t see them below.

View of object based on the type

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>null</td>
<td>null</td>
<td>a1</td>
</tr>
<tr>
<td>Animal</td>
<td>Animal</td>
<td>Animal</td>
</tr>
</tbody>
</table>

Components used from \( h \)

<table>
<thead>
<tr>
<th>h</th>
<th>a1</th>
</tr>
</thead>
<tbody>
<tr>
<td>age</td>
<td>6</td>
</tr>
<tr>
<td>isOlder (Animal)</td>
<td></td>
</tr>
<tr>
<td>getNoise()</td>
<td>toString()</td>
</tr>
<tr>
<td>getWeight()</td>
<td></td>
</tr>
</tbody>
</table>

Explicit casts: unary prefix operators

**Rule:** An object can be cast to the name of any partition that occurs within it — and to nothing else.

\( a0 \) can be cast to \( \text{Object} \), \( \text{Animal} \), \( \text{Cat} \).

An attempt to cast it to anything else causes an exception.

(Cat) \( c \)

(Object) \( c \)

(Animal) (Animal) (Cat) (Object) \( c \)

These casts don’t take any time. The object does not change. It’s a change of perception.

Implicit upward cast

**Call** \( c.\text{isOlder}(d) \)

Variable \( h \) is created. \( a1 \) is cast up to class \( \text{Animal} \) and stored in \( h \).

Upward casts done automatically when needed.

Type of \( h \) is \( \text{Animal} \). Syntactic property.

Determines at compile-time what components can be used: those available in \( \text{Animal} \).

If a method call is legal, the overriding rule determines which implementation is called.

Casting objects

You know about casts like:

\( \text{(int)} \) \((5.0 / 7.5)\)

\( \text{(double)} \) \(6\)

\(\text{Animal} h = \text{new Cat}("N", 5);\)

\(\text{Cat c = (Cat) h;}\)

A class cast doesn’t change the object. It just changes the perspective — how it is viewed!

Explicit casts:

- \( a0 \)
- \( a1 \)

Implicit upward cast:

- \( a0 \)
- \( a1 \)

Components used from \( h \):

- \( h.\text{toString}() \) OK — it’s in class \( \text{Object} \) partition.
- \( h.\text{isOlder}(...) \) OK — it’s in \( \text{Animal} \) partition.
- \( h.\text{getWeight}() \) ILLEGAL — not in \( \text{Animal} \) partition or \( \text{Object} \) partition.

By overriding rule, calls to \( \text{toString}() \) in \( \text{Dog} \) partition.
Explicit downward cast

```java
public class Cat extends Animal {
    private float weight;
    /** return true if o is a Cat and its
     * fields have same values as this */
    public boolean equals(Object o) {
        if ( ! (o instanceof Cat) ) return false;
        // { h is a Cat }
        if ( ! super.equals(o) ) return false;
        Cat c = (Cat) o ; // downward cast
        return weight == c.getWeight();
    }
    // getNoise() toString()
    getWeight()
}
```

Operator instanceof, explicit down cast

```java
public class Cat extends Animal {
    private float weight;
    /** return true if o is a Cat and its
     * fields have same values as this */
    public boolean equals(Object o) {
        if ( ! (o instanceof Cat) ) return false;
        // { h is a Cat }
        if ( ! super.equals(o) ) return false;
        Cat c = (Cat) o ; // downward cast
        return weight == c.getWeight();
    }
    // getNoise() toString()
    getWeight()
}
```

(Dog) o leads to runtime error.
Don’t try to cast an object to something that it is not!